THE MICROCOMPUTER MAGAZINE

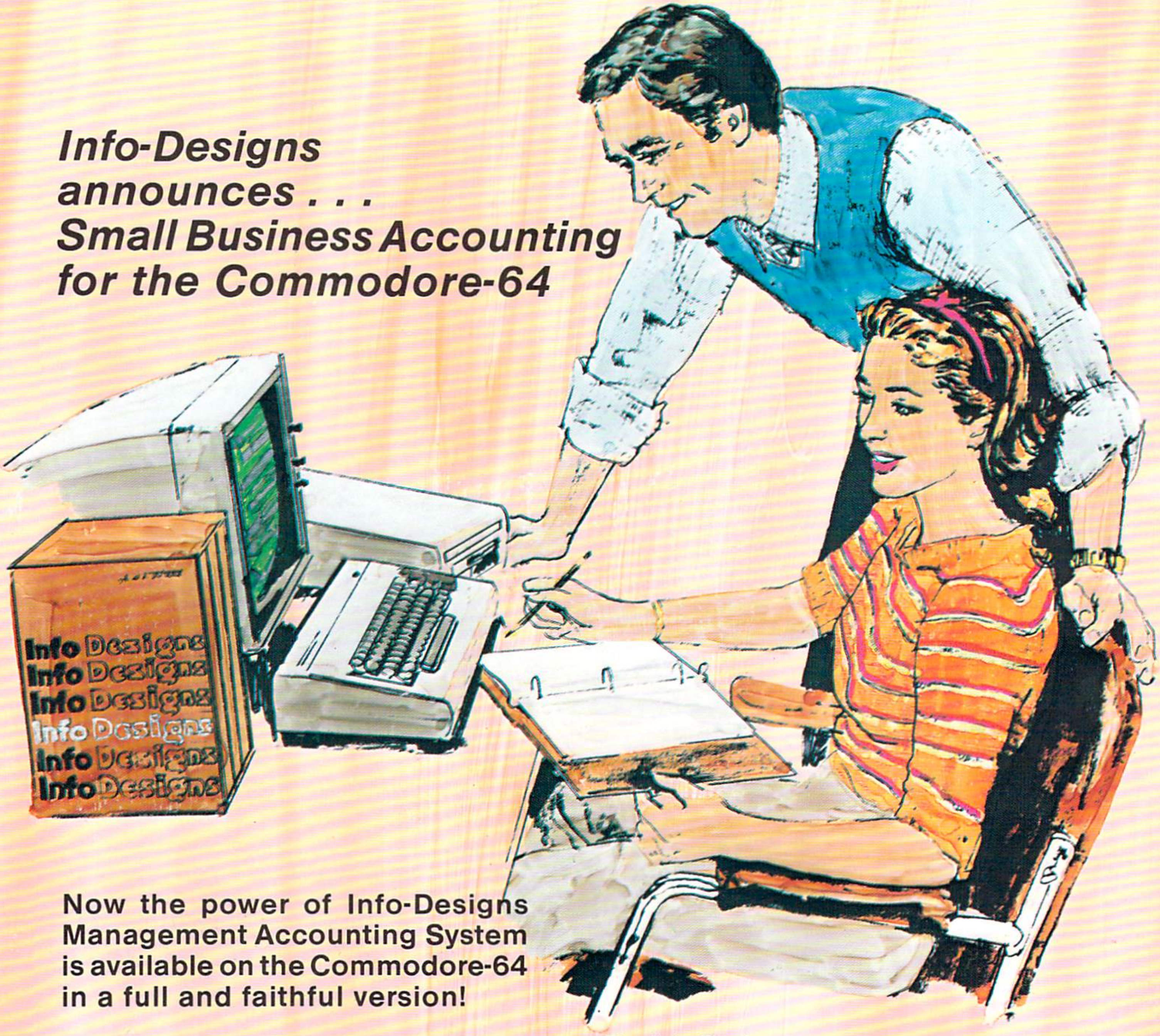
commodore

December 1982/January 1983
$2.50

- New C=64 Software
- Computerized House
- VIC Magic
- Go back to school with the Commodore 64

Commodore Computers in the Arts

Info-Designs announces . . . Small Business Accounting for the Commodore-64



Now the power of Info-Designs Management Accounting System is available on the Commodore-64 in a full and faithful version!

Thousands of these quality business accounting software packages have been sold on the CBM computer at $595 each. Now, similar features are available to the small business user on the new Commodore-64 for $199 per module!

Select the accounting modules you need—

- Accounts Receivable/Billing
- Accounts Payable/Checkwriting
- General Ledger
- Inventory Management
- Payroll
- Electronic Calendar ($149)

Limited Introductory Offer!

Our SoftPack combination contains the "Big-3" accounting—A/R, A/P and G/L—for only $495. Available for immediate delivery!

Flexible Design

The accounting system will work with one or two VIC-1541 disk drives (or 2031/4040 with IEEE interface), 1525 printer, and color or b&w monitor or TV.

Customer Support Plan

As part of Info-Designs ongoing effort to provide the highest quality microcomputer applications in the marketplace, we offer an optional telephone consulting service to support installation and ongoing operations.

Order NOW. . .for immediate delivery

See your local Commodore-64 Dealer or call us directly at **(313) 540-4010**. MasterCard and Visa accepted.

InfoDesigns

6905 Telegraph Road • Birmingham, MI 48010 • (313) 540-4010

Circle #1 on the Reader Service Card

THE MICROCOMPUTER MAGAZINE

commodore

DEPARTMENTS

commodore

Publishing Manager
Neil Harris

Editor
Diane LeBold

Staff Writers
Andy Finkel
Paul Goheen
April Koppenhaver
Keith Peterson
Michael Tomczyk

Contributing Writers
Bruce and David Cameron
Elizabeth Deal
Dale DePriest
Doris Dickenson
Mike Heck
George Kuetemeyer
Peggy O'Neal
Bruce Robinson
Joe Rotello
John Stockman

Technical Staff
Rick Cotton
Paul Goheen
Pat McAllister
Dave Middleton
Tom Rizol, Jr.

Circulation Manager
John O'Brien

Advertising Coordinator
Sharon Steinhofer

Advertising Manager
Diane LeBold

Commodore—The Microcomputer Magazine *is published bi-monthly by Commodore Business Machines, Inc., 487 Devon Park Drive, Wayne, PA 19087. Phone: (215) 687-9750. Domestic Subscriptions: six issues, $15.00. Application To Mail At Second-Class Postage Rates Is Pending At Wayne, PA 19087 and at additional mailing offices. POSTMASTER: Send address changes to* **Commodore—The Microcomputer Magazine,** *487 Devon Park Drive, Wayne, PA 19087.*

SEASONS GREETINGS AND BEST WISHES

to all our readers
from the staff at Commodore

Christmas screens for the Commodore 64, done completely with character graphics and sprites. Christmas trees in snow by Trip Denton of Philadelphia. All other screens by H. Rex Boucher, also of Philadelphia. Pop over to your nearest Commodore dealer to see the screens live, with music, as part of the great Commodore Christmas celebration.

Computer Art with the Commodore VIC 20

by
April M. Koppenhaver

Q: What do electronic sounds, fifty TV monitors, a department store, and the Commodore VIC 20 all have in common?

A: A presentation by Charles Cohen, Trip Denton, and Jeff Cain as part of John Wanamaker's celebration of "Art Week" in Philadelphia.

When you say "art", most people think of famous painters, writers, or musicians. Other people think of famous film stars, directors, or actors and actresses. Today however, "art" includes a relatively recent addition . . . Computer Art!

As technology becomes more and more sophisticatad, as well as more and more affordable, computers are becoming a major form of home entertainment and recreation. Several individuals have already given computers the necessary time, thought and consideration for becoming an outlet for human creativity and experimentation . . . above and beyond the ordinary usage and need for a computer. The innate human need to express something from within, to communicate a feeling or attitude, has led to the evolution of Computer Art.

The Commodore VIC 20 was featured in a two-hour Electronic Sound and Video Art performance at one of Philadelphia's most famous and prestigious department stores, John Wanamaker's. On October 5th, live electronic music was generated by Philadelphia musicians Charles Cohen and Jeff Cain on a variety of sophisticated synthesizers and electronic keyboards. The Commodore VIC 20 was directly integrated with the sound synthesizers to produce a textured variety of graphic and audio art.

The computer-generated artwork was created by video artist Trip Denton, also of Philadelphia. Most of the television sets and video screens found in Wanamaker's large Home Entertainment Department were involved in the visual presentation. (Imagine the experience of viewing approximately fifty television screens that are displaying color patterns and designs, in synchrony with electronic music . . . wild!)

Each of the three individuals who performed are professionals. Charles Cohen and Jeff Cain are sound designers and electronic composers. They have composed music for the National Public Radio show, "Visions of the Future", and were featured in NPR's "Electronic Minstrels" documentary. Cohen and Cain record under the name of "Ghostwriters" and have recently completed their second album on Zero Records, entitled "Objects in Mirrors".

Charles Cohen holds a BS degree in Liberal Arts from Arizona State University, and a Masters in Fine Arts from Temple University School of Communications and Theater. For the past thirteen years, Cohen has been the sound designer for the theater department at Temple University. In this capacity, he is responsible for creating 'environments of sound' along with the actual creation of sound.

In a recent interview, Cohen stated that he would hesitate to call himself a musician. Instead, he expressed his art as "using pure sounds which are intuitively arrived at, to create a new medium." Cohen said that he liked working with computers to make sounds, as well as using computers as controllers for an analog synthesizer. Cohen's most admired electronic composer is Morton Subotnick, the 'Frank Sinatra' of electronic music. His dream is to hire about five incredible musicians to perform electronic sound. Past applications of his efforts were incorporated in a variety of modern dance exhibitions, theater experiments, and video art tapes.

As far as introducing a brand new art, Cohen feels Philadelphia offers a cross-fertilization environment where he can gain valuable experience and exposure. Finding other artists interested in similar endeavors is always a plus. Cohen has been working with Jeff Cain on specific events over the past eleven years, but only recently

met Trip Denton and connected with what he was doing and creating. Cohen then invited Trip to participate in the John Wanamaker's presentation . . . the rest is history.

Trip Denton graduated from the Pennsylvania Academy of Fine Arts with a degree in painting. Trip's artistic talent is seen in his many finely designed video screens. Denton's interest in computers has followed him since junior high school days. However, only recently, since computers have become affordable, has Denton been able to pursue these interests.

Denton researched the market to find what was the best computer for the price. His decision to buy Commodore was not a hard one. "For the price and performance there is no comparison with Commodore equipment. I liked the fact that the VIC has a standard keyboard, not just a membrane. I like the fact that I was able to do color graphics", said Denton during our interview. "I have been more than satisfied with VIC's performance. Also, I can't express how much I appreciate Commodore's support and interest in Charles' and my applications.

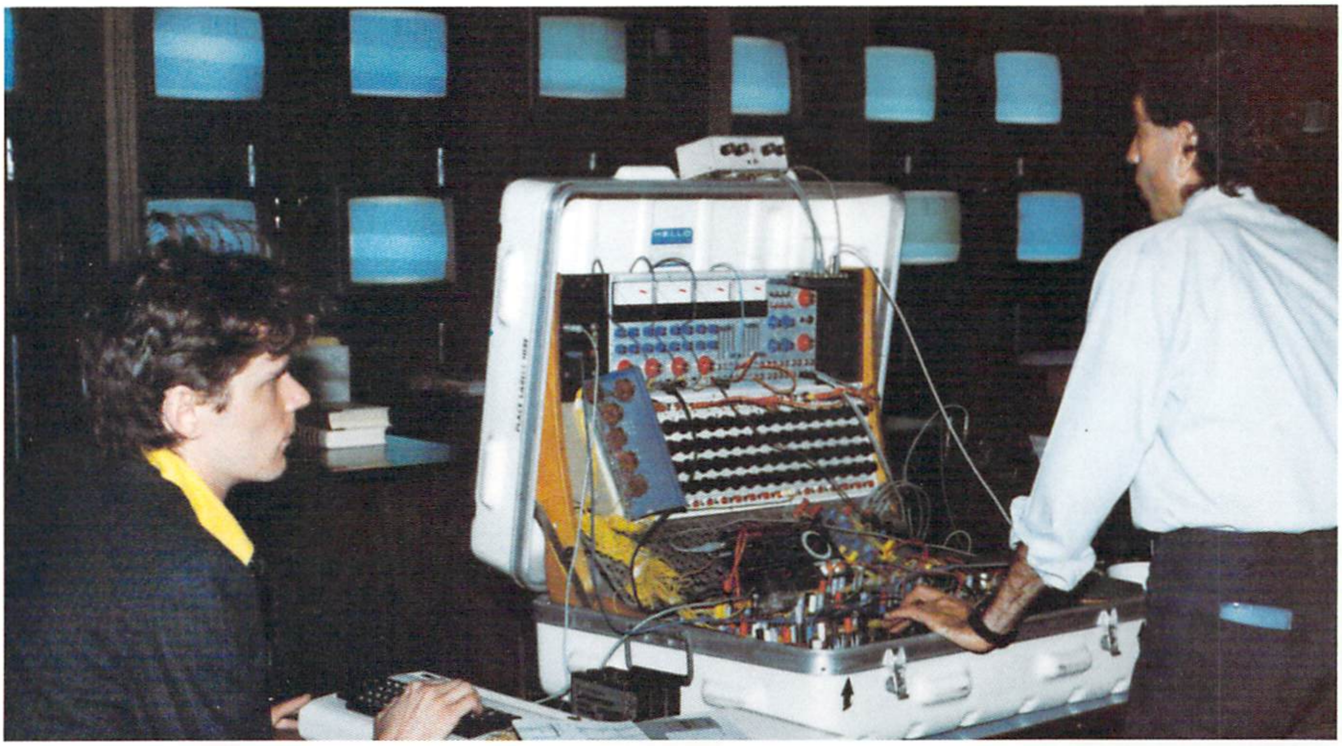
Jeff Cain, Philadelphia computer musician, at his synthesizer keyboard.

Charles Cohen (l) sends electronic signals to Trip Denton's VIC 20 graphics program while coordinating music and graphics at a Philadelphia celebration of computer art.

"A work of art is usually described by a set of rules or parameters. Painting on a canvas is more precise than displaying images through the use of a computer. However, the immediacy of a computer allows you to change colors and designs much more rapidly . . . besides, you never have to wait for the paints to dry."

Denton also feels the computer will be a more recognized tool for the artist in the next few years. As far as when a computer course will become a requirement for art students, Denton would not comment. However, he did express surprise with how easy it was to actually operate and write programs for the VIC. He summarized with, "I strongly recommend that anyone with a similar desire to express art by using a computer should not hesitate to buy Commodore. My experiences have always been good ones."

Commodore takes great pride in being a part of this revolutionary concept of Computer Art.

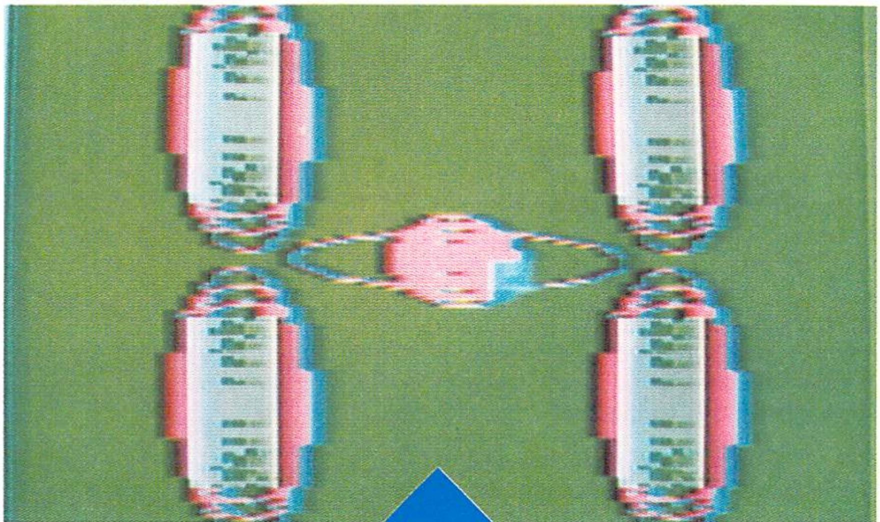


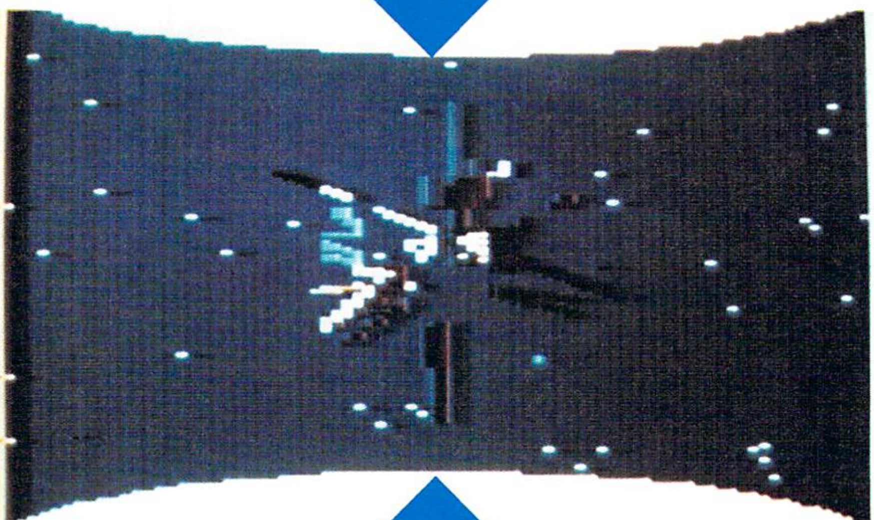
Trip Denton (l), computer artist, and Charles Cohen, computer musician, begin the VIC 20 sound presentation at John Wanamaker's computer art celebration in Philadelphia. Every TV screen in Wanamaker's home electronics department carried the graphics display.

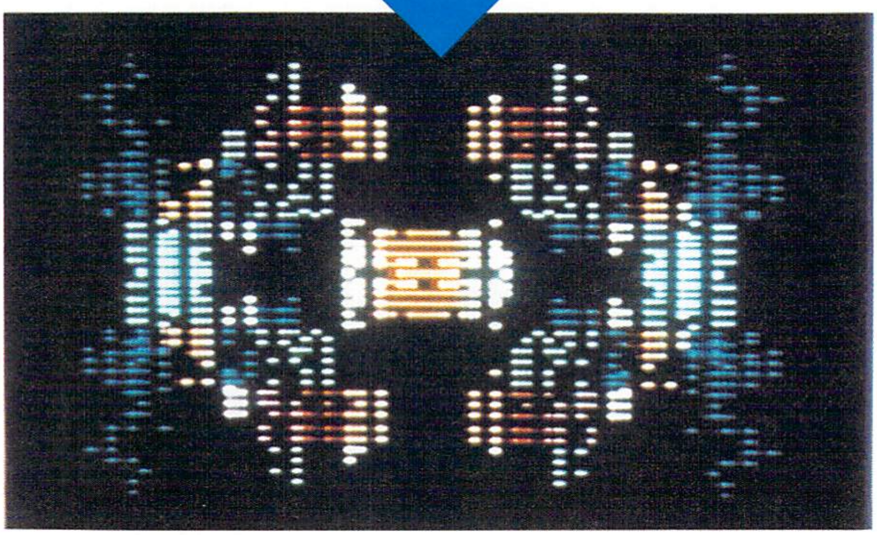
Creating Your Own Computer Graphics

With the help of the VIC 20 Super Expander and this little program, you, too, can create the kinds of graphic designs you see on our cover and on these pages. Thanks to Trip Denton for providing it for our readers.

```
 1 DIMD(21)
 2 GOSUB39
 3 GRAPHIC3:RF=1:C=2
 5 COLORINT(RND(1)*15),INT
   (RND(1)*7),INT(RND(1)*7),
   INT(RND(1)*15)
 6 J=INT(RND(1)*1024):
   K=INT(RND(1)*1024)
 7 IFRCOLR(1)=RCOLR(2)THEN5
 8 IFRCOLR(2)=RCOLR(3)THEN5
 9 J=511:K=511
10 X=J:Y=K
11 A=X:B=Y
12 FORG=1TO20
13 R=10*RF
14 N=D(G)
15 ONN+1GOTO24,17,16,19,18,
   21,20,23,22
16 X=X+R
17 Y=Y-R:GOTO24
18 Y=Y+R
19 X=X+R:GOTO24
20 X=X-R
21 Y=Y+R:GOTO24
22 Y=Y-R
23 X=X-R
24 IFX>1023THENX=1023
25 IFX<0THENX=0
26 IFY>1023THENY=1023
27 IFY<0THENY=0
28 DRAWC,A,BTOX,Y
29 GETA$:IFA$>""THEN:
   REGIONINT(RND(1)*16):
   IFINT(RND(1)*3)=1THEN2
30 C=C+1:IFC=4THENC=2
31 A=X:B=Y
32 NEXT
35 RF=RF+1:IFRF=40THEN37
36 GOTO6
37 FORT=0TO333:NEXT
38 GOTO2
39 FORM=1TO20
40 Q=INT(RND(1)*9)
41 D(M)=Q
42 NEXT
43 RETURN
READY.
```



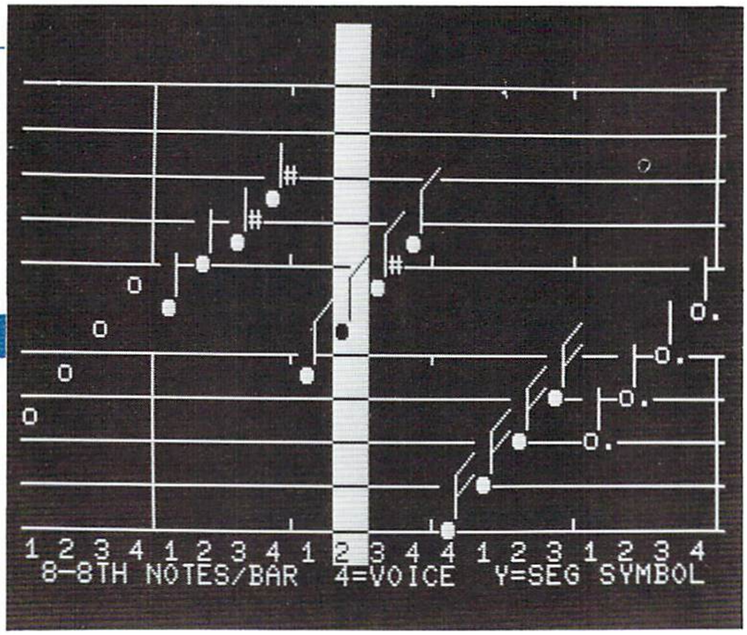




VIC 20 graphics designed by Trip Denton, Philadelphia computer artist.

The History of Computer Music, *Commodore-Style*

by Neil Harris



Everyone today knows that computers make music. Synthesizers are a staple in rock bands, and computer sound effects are heard in movies and video arcades. It's hard to remember back to a time when computers were only used for boring applications like accounting and guiding missiles. But even in those ancient days, imaginative computerists were getting their systems to sing.

The first computer music I ever heard was on a BASIC/4 business computer. One mad genius programmer spent all his spare time programming John Philip Sousa marches into a room-full of computer equipment. The printers provided the percussion, typing along in syncopated rhythms, along with tinkling sounds from their bells. Video terminals played melody and harmony with different pitches of beeps and bloops. The bass part came from the magnetic disk platters, which were somehow made to rumble along. You can't imagine what ''Stars and Stripes Forever'' sounds like on an orchestra like this!

During the first year that Commodore's PET was available, two discoveries made computer music possible. One was easy, and the other so peculiar that it's never been seen since.

The weird way was shown to me in the Mr. Calculator store that Commodore used to have in Philadelphia. A complete stranger, whom I've never seen since, placed a transistor radio on top of the keyboard. He ran a machine language program into the machine, and then tuned the radio carefully. When he found the right station, you could hear a staticky and buzzing version of Scott Joplin's ''The Entertainer''. The song was created by maximizing the Rf interference generated by the computer. The FCC would have been appalled.

That same year, an article appeared in a user group publication called *Pet User Notes,* published by Gene Beals. It described a method by which the 6522 input/output chip made tones that could be amplified and fed into a speaker. It was much simpler than any other method. All it took was three POKEs: one to start the chip in free-running mode, another to set the waveform and octave, and the third to set the actual frequency of the note.

The music was available from the CB2 line of the user port in back of the PET, and was henceforth called CB2 sound. A description of CB2 sound with sample programs is contained in insert 1 of this article.

CB2 sound was fun as a toy, but not really capable of creating true music. For one thing, there was no capability for harmony, since there was only one tone generator. For another, the waveform was limited to square waves, a fairly unnatural sound.

The next major innovation was perfected by Hal Chamberlain at a company called Micro Technology Unlimited (or MTU). He came up with a digital-to-analog board that plugged into the user port and cassette port, and a special machine language routine to make it work. With Chamberlain's board, an amplifier, and some machine language software, four-part harmonies could be played.

The real advantage to Chamberlain's system was that the waveforms were completely programmable. A waveform table is set up in RAM containing 256 bytes of information. It takes all 256 bytes and shifts them through a register at whatever frequency desired. The values of the bytes determined the amplitude of the wave at that instant in time. A sample waveform is shown in Chart 1. You could now get sounds from the computer like real instruments: guitars, violins, pianos, and even percussion.

People could play canned programs on their system, like the very fine collection programmed by Lou Cargill for AB Computers (including all kinds of music: jazz, classical, disco, and blues) and the public-domain programs by Frank

Covitz collected in the Toronto PET User Group's library. Combined with computer graphics, you could put on quite a show.

Unfortunately, not all of us are machine language geniuses like Hal Chamberlain. Even though he provided an enhanced machine language monitor for editing music, very few people could create music themselves with this system.

The next breakthrough was the creation of the first "user-friendly" software system for composing music, the "Visible Music Monitor" by Dr. Frank Levinson, also sold by AB Computers (along with their inexpensive KL-4M music board). This program puts the musical staves on the screen, and lets you use the editing keys of the PET to create and edit a musical score. It includes some sophisticated functions, like key transpositions, octave changes, and multiple waveforms, but is simple enough for anyone to use. Sheet music is typed into the computer very easily. Sections can be repeated, making the music extremely memory efficient.

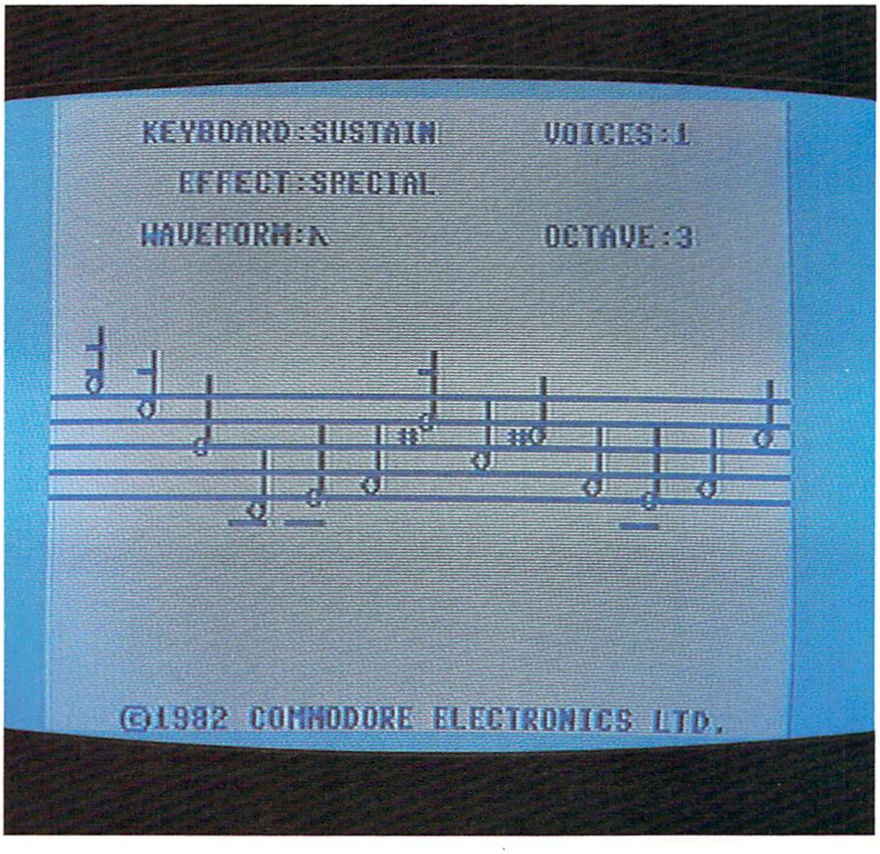
At last, average users could compose and transcribe music for their own computers.

Now we come to the VIC 20. The VIC contains the same 6522 I/O chip that made PET music possible, but it also has the VIC chip with its own four-part music capability. Through simple POKE commands in BASIC, anyone could create music and sounds, by controlling the volume and pitch. Even the user manual has programs for music that would have been very complex just a few short years ago.

The computer music world keeps evolving. This year Commodore released the new Commodore 64. The 64 uses a special chip called SID just for computer music. The SID chip is a complete synthesizer on one chip. In fact, Bob Yannes, the chip designer, took a commercial $750 synthesizer and put every one of its functions on one chip. The SID has controls for volume, frequency, waveform (attack, decay, sustain, and release are independently programmable), filters, ring modulation and pulse generation. A complete musical instrument on a chip!

To complement the music capabilities of this chip, Commodore is designing two pieces of software: a music composer cartridge and a musical instrument cartridge. "The Composer", written by Andy Finkel, lets you enter sheet music graphically, something like the old "Visible Music" program. Paul Higginbottom's musical instrument program lets you use the Commodore 64's keyboard just like a synthesizer, creating rhythms and different sounds. Choose your program and you could be the next Bach (or the next Devo).

Where is computer music going from here? There is a whole world of possibilities. How would you like your computer to sing along with the music it makes? Commodore's new Speech Technology Division could make that a reality within a short time. Maybe we'll get to the point where the computer composes by itself!



Now that you've had an overview of Commodore computer music, we're inaugurating a regular column on this subject. Writing for us is Lee Silvan, a professor of music at Community College of Philadelphia, author of textbooks on music and owner of a houseful of computers. Look for his first column in the next issue of this magazine.

References:

AB Computers
252 Bethlehem Pike
Colmar, PA 18915
PET User notes . . . $1 apiece
Visible Music Monitor . . . $35
Music exchange for VMM . . . $1 each (write for list)

Micro Technology Unlimited
2608 Hillsborough Street
PO Box 12106
Raleigh, NC 27605
Music Board . . . $84
Instrument Music Software . . . $50

Toronto PET Users Group
381 Lawrence Avenue West
Toronto, Canada M5M 1B9
Membership . . . $20, write for software list

How To Use CB2

CB2 music works in any Commodore computer with a 6522 chip (PET, CBM, VIC, but not 64 and Max). To play CB2 music on a PET or CBM, type the following POKEs (see the VIC Reference Guide for 6522 locations):

POKE 59467,15: REM set free-running mode
POKE 59466,X: REM X=15 octave, X=51 middle, X=85 highest
POKE 59464,N: REM N=frequency of note

This will result in a square wave output, where the frequency can be determined from the following equation:

$$\text{Frequency} = \frac{500{,}000\text{ Hz}}{(N+2)*(Y)} \quad \text{where } Y=2 \text{ when } X=85,\; Y=4 \text{ when } X=51,\; Y=8 \text{ when } X=15$$

To turn off square waves, just POKE 59467,0

*The square waves are produced through the built-in speaker of the CBM, and through a pin of the User Port. You can hook up a speaker and amplifier to this pin (you can get a cheap one for about $10 from R*d** Sh*ck). Hook up one wire from the amp to CB2, and the other wire to a ground (see the pinout diagram in your manual).*

```
10 REM CB2 KEYBOARD FOR PET, BY ---> NEIL HARRIS
20 REM USE KEYS TO PLAY NOTES
30 REM SHIFT CHANGES OCTAVE
90 POKE59467,16
100 DATA32,251,48,244,24,237,40,224,31,211
105 DATA47,199,23.188,39,182,30,177,46,167
110 DATA22,157,38,149,29,140,45,132,21,124
115 DATA37,120,28,117,44,111,20,104,36,99
120 DATA27,93,59,90,26,87,42,82,18,77
125 DATA34,73,25,69,41,65,17,61,33,59
130 DIMA(80):FORL=1TO30:READA,B:A(A)=B:NEXT
140 J=1:K=85
145 IFPEEK(152)<>JTHENJ=1-J:K=100-K:POKE59466,K
150 Q=PEEK(151):IFQ=255THENQ=0:IFI=1THEN145
155 IFQ=65THENI=1-I:GOTO145
160 IFQ=80THENFORL=1TO9:GETA$:NEXT:POKE59467,0:EN
165 IF(PEEK(59457)AND4)<4THEN180
166 IF(PEEK(59457)AND16)<>16THEN200
167 IF(PEEK(59457)AND1)<>1THEN220
170 POKE59464,A(Q):GOTO145
180 FORL=A(Q)TO0STEP-5:POKE59464,L
185 NH=PEEK(59457):IFNH<>255ANDNH<>127THENNEXT
190 GOTO145
200 FORL=A(Q)TO255STEP5:POKE59464,L
205 NH=PEEK(59457):IFNH<>255ANDNH<>127THENNEXT
210 GOTO145
220 W=Q+16:IFW>48ANDW<>59THENW=Q-16:GOTO230
225 IFW>40THENW=W-7:GOTO230
227 W=W+8
230 POKE59464,A(Q):FORL=1TO10:NEXT
235 POKE59464,A(W):FORL=1TO5:NEXT:GOTO145
```

```
10 REM CB2 WOLF WHISTLE, BY ---> NEIL HARRIS
100 POKE59467,16:POKE59466,15
110 FORL=180TO76STEP-3:POKE59464,L:NEXT
120 FORL=200TO100STEP-3:POKE59464,L:NEXT
130 FORL=100TO250STEP3:POKE59464,L:NEXT
999 POKE59467,0
```

```
10 REM CB2 ALLEY KAT, BY ---> NEIL HARRIS
90 X=200
100 DATA117,1,104,2,93,1,90,2,93,2,104,2,111,2,117,2,111,2
110 DATA104,4,117,2,117,1,111,2,104,2,99,2,93,6
120 DATA117,1,104,2,93,1,90,2,93,2,104,2,111,2,117,2,111,2
130 DATA104,4,117,2,117,1,111,2,104,2,93,2,90,6
140 DATA90,2,90,1,81.1,81,1,81,3,90,2,81,4
150 DATA90,2,90,1,81,1,81,1,81,3,90,2,81,4
160 DATA90,2,90,1,81,1,81,1,81,3,90,2,81,4
170 DATA90,2,90,1,81,1,81,1,81,1,90,2,90,1
180 DATA93.1,93,1,93,1,104,2,104,1.117,1,117,1,117,1,132,2
400 DATA132,1,140,2,999,0
500 POKE59467,16:POKE59466,15
510 READA,B:IFA=999THENRESTORE:X=X*.75:GOTO510
515 GETA$
516 IFA$<>""THENPOKE59467,0:STOP:POKE59467,16:POKE59466,15
520 POKE59464,A:FORL=1TOX*B:NEXT:POKE59464,0:GOTO510
```

Dealers —
We are the exclusive source for the oldest accounting system exclusively for Commodore. See our February ad for more information, including the all-new inventory module.

CMS GENERAL ACCOUNTING SYSTEM

A fully interactive General Accounting System designed especially for the first time user. All input requests are fully prompted with complete verification of input data. Most reports may be printed either to the screen or the printer and started or stopped at any point. The user is led completely through each function by a series of highlighted prompts fully explaining the required input at each point. A professionally written instruction manual is included which shows sample reports generated by the system and further explains each step and prompt as it is encountered by the user. These user prompts, together with the detailed step by step manual, make the system extremely user friendly.

GENERAL LEDGER

Up to 1000 accounts on the Chart of Accounts. Fully departmentalized up to ninety-nine departments. A Cash Receipts and Disbursements Journal allows each check or deposit to be distributed to nine separate General Ledger accounts. Non cash postings may be entered through the General Journal. The Cash Journal and General Journal may be printed at any time either to the screen or to the printer. A non structured Chart of Accounts allows easy tailoring of the Financial Statements to fit varying user needs. Account balances are maintained for current month, quarter to date, and year to date. A detailed Budget Analysis Report may be printed at any time either to the screen or to the printer. The General Ledger program may be used alone or may be set up to accept postings from Accounts Receivable, Accounts Payable, Payroll, or other programs.

ACCOUNTS PAYABLE

Allows a total of 2500 invoices and vendors per 8050 data disk. Each invoice may be distributed among nine separate General Ledger expense accounts. Prints checks with complete voucher detail for up to thirteen invoices per check. Invoices may be paid by vendor range or by individual invoice numbers. Credit memos and term discounts are taken automatically. Full or partial payments may be made for each invoice. A Check Register may be printed at any time. Complete aging reports with user set aging breaks may be printed at any time either to the screen or to the printer. A special Cash Requirements Analysis Report is available to assist in determining upcoming cash requirements. The Accounts Payable program may be used alone or may be set up to automatically update the General Ledger.

ACCOUNTS RECEIVABLE

Allows a total of 1800 invoices and customers per 8050 data disk. Each invoice line item may be distributed to a separate General Ledger income account. Prints invoices and monthly statements and allows for individualized messages on the statements. Finance charges may be automatically added to overdue invoices at a user determined rate and period. Complete invoice aging reports with user set aging breaks may be printed at any time either to the screen or to the printer. A special Overdue Invoice Report is also available for close monitoring of receivables. Handles credit memos as well as invoices. A Sales Tax Report is available showing total sales and sales taxes payable for up to nine separate sales tax rates. Total sales and sales commissions earned are automatically tracked for up to fifteen salesmen. The Accounts Receivable program may be used alone or set up to automatically update the General Ledger.

PAYROLL

Maintains current, quarterly, and yearly totals for 350 employees. Prints Payroll checks with full deduction and pay detail as well as year to date totals. Accommodates weekly, bi-weekly, semi-monthly, and monthly employees. Pays regular, overtime, holiday, and piecework hours. Handles federal and state deductions plus up to eight miscellaneous deductions and payments per employee. Prints Payroll Journal, Check Register, and an Absentee Report as well as state and federal 941 information and W-2 forms. The Payroll program may be used alone or set up to automatically update the General Ledger.

Circle #4 on the Reader Service Card

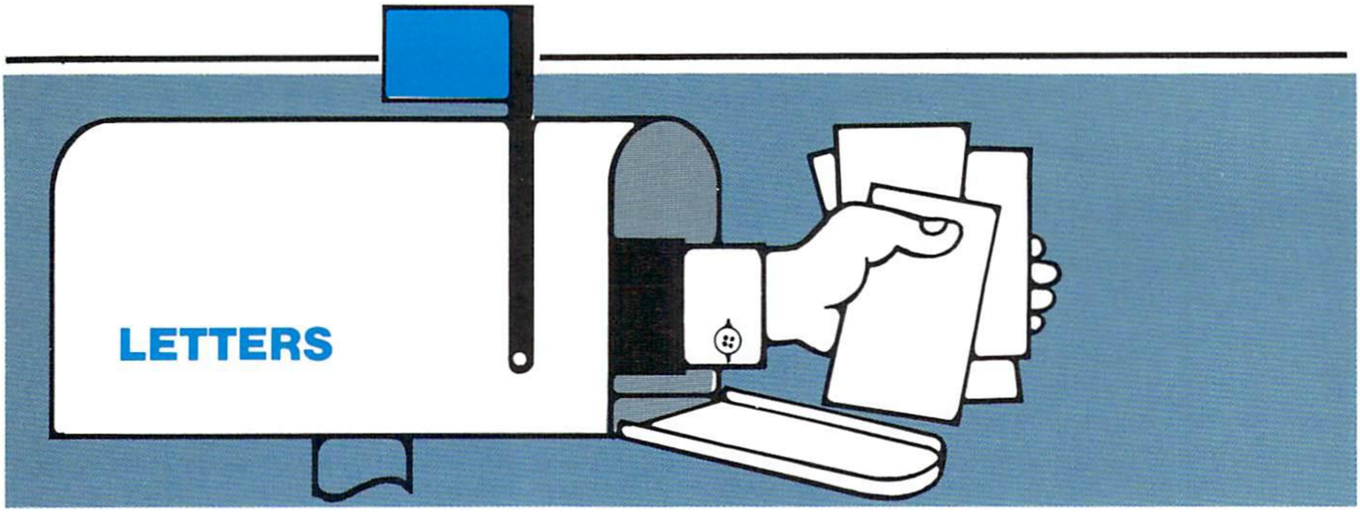
W E

Southern Solutions

P.O. Box P, McKinney, Texas 75069 - (214) 542-0278

— We have 11 programs for the 64! —
Call for details

OFFICIALLY APPROVED BY COMMODORE



September 22, 1982

Dear Editor,

I would like to thank your magazine for designating me as a software educational resource center person as mentioned in the August/September issue of *Commodore*.

In this same issue appeared two articles by Dwight Wheeler on "Some other Sorts". I have enclosed a copy of the two sorts he spoke of with very minor changes.

In both the "Bubble Sort" and the "Shell Sort" Mr. Wheeler wrote about, if one were to merely omit the string symbol "$" the sorts will arrange numerical data in ascending/descending order.

I have indicated in both attached programs line numbers 231 and 232 how one would easily alter line 240 (in both programs) to yield the order desired.

I use these sorts often in my teaching efforts here in the Rapid City School System and in mathematics especially, the numerical sortings are quite useful.

I generally wouldn't write to a magazine about this, but I felt that your readers might consider the above mentioned omissions for their classroom use, etc.

I subscribe to many magazines in the micro field, but I especially like yours for their programs. Our school system uses almost entirely the PET's. We have two Trash-80's, however. As a member of the micro selection committee, I was the driving force behind the purchase of the PET's over the Apples. It's tough to teach the piano without one and equally difficult to teach computing without hands on experience.

Sincerely,
Jim L. White

```
100 REM >> PGM: SSORT (NUMERICAL) <<
110 DIM A(100)
120 REM .... LOAD ARRAY ....
130 N=1
140 READ A(N)
150 IF A(N)=9999 THEN 180
151 REM .... USE A LARGE ENOUGH NUMBER
152 REM .... TO END PROGRAM
160 N=N+1
170 GOTO 140
180 N=N-1
190 REM .... SORT ....
200 G=N/2
210 IF G=0 THEN 350
220 FOR I=1 TO N-G
230 IF A(I) <= A(I+G) THEN 290
231 REM .... > IS DESCENDING
232 REM .... < IS ASCENDING
240 REM .... EXCHANGE ....
250 T = A(I)
260 A(I) =A(I+G)
270 A(I+G)=T
280 E=I
290 NEXT I
300 IF E=0 THEN 330
310 E=0
320 GOTO 210
330 G=INT(G/2)
340 GOTO 210
350 REM .... PRINT ....
351 CLOSE4
360 FOR X=1 TO N
370 PRINT A(X);" ";
380 NEXT X
390 END
400 DATA 78,45,12,89,56,23,7,4,1,8,5,2,9,6,3,9999

READY.

 1
 2
 3
 4
 5
 6
 7
 8
 9
```

```
 12
 23
 45
 56
 78
 89

READY.

 100 REM >> PGM: BBSORT (NUMERICAL)
 110 DIM A(100)
 120 REM .... LOAD ARRAY ....
 130 N=1
 140 READ A(N)
 150 IF A(N)=9999 THEN 180
 151 REM .... USE A LARGE ENOUGH NUMBER
 152 REM .... TO END PROGRAM
 160 N=N+1
 170 GOTO 140
 180 N=N-1
 190 REM .... SORT ....
 200 B=N
 210 IF B=0 THEN 330
 220 J=0
 230 FOR I=1 TO B-1
 231 REM .... > IS DESCENDING
 232 REM .... < IS ASCENDING
 240 IF A(I) >= A(I+1) THEN 290
 249 REM .... EXCHANGE ....
 250 T=A(I)
 260 A(I) =A(I+1)
 270 A(I+1)=T
 280 J=I
 290 NEXT I
 300 B=J
 310 GOTO 210
 320 REM .... PRINT ....
 330 FOR X=1 TO N
 340 PRINT A(X);" ";
 350 NEXT X
 360 END
 370 DATA 78,45,12,89,56,23,7,4,1,8,5,2,
      9,6,3,9999
READY.

 89
 78
 56
 45
 23
 12
 9
 8
 7
 6
 5
 4
 3
 2
```

EDITOR'S NOTES

Now that Neil Harris has stepped in as our new Publishing Manager, you'll notice a few changes in this issue. First, we're heavier than usual in programmer's tips. We think this will be a welcome change for our many "hackers" (otherwise known as "hobbyists") out there. Since Neil is a programmer himself, you can expect that emphasis to continue, although we'll by no means be neglecting other segments of our audience. Besides, we feel pretty certain those "other segments" will also enjoy the additional programming tips. If you've got the computer, you may as well get the most out of it, right?

You'll also notice a bit of change in the look of the book. Our plan is to keep upgrading the appearance—as well as the quality of the information—to make it easier on your eyes, more enjoyable to read, and easier to locate the information you want.

If you have any doubts concerning Neil's dedication to giving our readers the most and best information for their money, doubt no more. Just take a look at his "From Where I Sit" column on page 15.

Meanwhile, back in the editor's office—which is becoming increasingly crammed full of papers, disks, tapes, photos, equipment and silly sayings posted on the walls—let me point out some of the highlights of this issue. The great versatility of our VIC 20 is demonstrated once again in April Koppenhaver's story on page 4. A group of inventive Philadelphia artists and musicians are using the VIC to create computer graphics and then to coordinate the visual effects with synthesized music. The results are very interesting.

You'll also be glad to see the review of the *Commodore 64 Programmer's Reference Guide* that we promised last issue. As Mike Heck points out in that review, the *64 PRG* is a fully detailed, extremely useful tool for sophisticated programmers, yet takes the time to explain things for beginners.

Also on the 64—which will undoubtedly remain the hot item in the computer marketplace for some time to come—is Neil Harris' review of Commodore's *EasyScript* word processor. A relatively inexpensive system, *EasyScript* nevertheless packs extraordinary power that exceeds some of the best professional-quality systems presently in use. At the risk of sounding a little evangelical, we're convinced that this is only the beginning for the 64, and you'll be seeing an unprecedented wave of applications before 1983 is very far along.

A topic that interests me personally—speech technology—is only touched on in this issue's Commodore News department. But you can be sure you'll be hearing more about that in the future, especially as our new speech technology division begins to achieve its early goals.

Before I close, on behalf of the whole gang here at Commodore, let me wish you the best holiday season ever and a full and happy 1983!

Diane LeBold

Diane LeBold
Editor

FROM WHERE I SIT

It was March of 1978 when I first heard of Commodore. There was an ad in the local newspaper for a "salesman for a computer store". I had never heard of a computer store before, but I needed a job, and had both programming and sales experience.

I made an appointment with Gene Beals, manager of the store, which was called Mr. Calculator. I arrived at 1 p.m. for my appointment. The store, all of 15 by 15 in size, was jam-packed with customers. About half the people were buying discount calculators in response to a full-page ad in the Sunday paper, and the other half were looking at a Commodore PET 2001 personal computer.

Gene was so busy running from one customer to the next, selling calculators and looking haggard, that he didn't have time to pay attention to the computer. One of the customers typed in her name, and hit the RETURN key, only to be greeted by the message SYNTAX ERROR. I pushed in close to the computer and typed a PRINT statement with her name, which appeared on the screen.

The crowd gasped.

Feeling a little like a showman, I tried a few other simple BASIC programs, which, Lo! and Behold! worked with no problem. Picking up the manual, I learned a little about what the machine could do. The rest of the afternoon was spent playing with the computer and answering people's questions as best I could.

At 6:30, when the store closed for the day, Gene finally had time to interview me. At 10 a.m. the next day, I got a phone call at home, with the message, "When can you start?" A career was born.

It was a time for pioneers. There were no accessories or software for the product yet, just the computer with a built-in screen and cassette deck. The software

by Neil Harris

and manuals weren't ready at that time either. We learned new things about the computer every day, like the music and sound capabilities (see my article later this issue) and the existence of abbreviations for the BASIC commands.

Since then, I've worked in other microcomputer sales operations, both retail and mail order, and also had business programming jobs. Around the end of 1980, I started hearing rumors about a new Commodore computer, one that would sell for $300 with color, sound and full PET BASIC. I knew right away that, if true, the product would sell more than any other computer ever. I called Commodore at their new headquarters in Valley Forge, and was told to talk to Mike Tomczyk about the new machine, which was to be called the VIC 20. I never did manage to reach Mike on the phone, though.

Finally, in January of 1981, I heard a radio announcement of an open house at Commodore. I always liked Commodore computers better than the other brands, because of the features like the screen editor that made it easy to program and fix mistakes. So I went on my day off, and spoke to the personnel director. I filled her in on my background, which included some articles written for *COMPUTE!* magazine. She immediately decided that Mike Tomczyk was the one I needed to speak with.

Fortunately, Mike was astute enough to give me the job. Since then, I've had the privilege to work with the VIC since its inception, co-authoring the user manual and reference manual, writing software, and helping with sales support. Now I'm even more lucky to be working with the Commodore magazines.

I'm using my technical and marketing perspective to help continue the improvement in our publications. During the next few issues, you'll see increasing coverage of the programming side of our machines, to go along with the business and education features. The magazines will provide an inexpensive source of software, with more programs each issue. You will begin to find this magazine and the sister publication, *Power/Play,* on the newsstands and in bookstores.

I am a firm believer in the Commodore "religion". This has come from my continual contact with people, both experienced programmers as well as beginners, who absolutely love working with our computers. It is you out there in user-land who will provide this magazine with articles and programs that make it easier for everyone to get the most out of your Commodore computers.

Don't worry if you don't have writing experience. We have editors who can clean up your article for publication, if you have something interesting and worthwhile to communicate. Remember, this is a *user* magazine.

So send in your letters, comments, software, and articles. My address is: Commodore Business Machines, 487 Devon Park Drive, Wayne, PA 19087

Commodore's New Speech Technology Division Aims at Consumer Users

In an exciting step into the future, Commodore has established a new speech technology division located in Dallas, Texas. Under the supervision of voice processing expert Richard Wiggins, the new division is focused primarily on making computer voice input and output affordable to Commodore's home computer users. In the future, the division will be working on more complex business-oriented projects, as well.

The division's first goal is to create voice output for Commodore home computers, to allow the computer to "speak" to the user. According to Wiggins, one of the major functions of voice output is to let the computer interact with people who don't read—primarily small children.

"This opens many new possibilities for educating small children—and gives the computer a new dimension," Wiggins said, then expanded further on the potentials. "Voice output can also be used to teach spelling and languages to both children and adults, or can be used in any game or activity where you have a lot of information on the screen already.

"That dimension takes us rapidly into business applications. It's not hard to imagine a word processor in which the computer lets you know when you're nearing the end of the page, or have misspelled a word, for instance," he added.

More immediately, the Commodore speech technology experts are working on programming game ROMs for voice output.

"You collect samples of the type of voice you want and program that voice into the computer, using vocal track models that reproduce the sound-making of humans through digital filtering. That information can be stored in a cartridge, to, say, project a certain image in a game," Wiggins explained further. "The computer can also be programmed to respond vocally to words typed in by the user."

According to Wiggins, reliable voice input—in which the computer responds to vocal commands given by the user—is harder to achieve.

"In most voice-input systems that exist right now, you have to control what you say—speak in a very structured format—for the computer to understand you," Wiggins went on. "You must be very careful. For instance, people in the United States have different accents, which means they could pronounce the same word differently. Or take the words like 'red' and 'read' that are pronounced the same but have different meanings. If you aren't careful and the computer misunderstands a word, what if that led to something disastrous happening in the program?"

Right now, he said, machines are available that can adapt to a particular person's voice by collecting voice patterns. That type of speech processing is called speaker recognition or speaker verification, and can be very useful for various security procedures.

Whatever the voice input or output is used for, however, the number of words you enter into the computer, and what they are, has a direct relationship to the cost of the product, Wiggins explained. As time goes on, he said, Commodore's speech technology division will be interested in developing large-vocabulary, high-performance systems as well as consumer-oriented, lower-vocabulary products.

Circle #7 on the Reader Service Card

Circle #8 on the Reader Service Card

Commodore 64 Software *Update*

If you haven't seen it all, yet, at your Commodore retailer, you can be sure you'll be seeing it soon—a whole line of software for the Commodore 64 that will undoubtedly exceed your performance expectations—and costs a lot less than comparable products.

We've reviewed the *EasyScript* word processor and *Word/Name Machine* word processor/mail list in detail on page 30. As Neil Harris points out in that review, *EasyScript* is a professional-quality word processer that does everything you'd want it to—and more. The *Word Machine* word processor is a simpler product, intended for home use, that is ideal for short documents, such as letters or notes. Used in combination with the *Name Machine* mail list, the *Word Machine* can even generate personalized form letters.

You'll also find more details about Commodore's 656 public domain educational programs available for the 64—and the rest of the Commodore line as well—in this issue on page 24 Commodore has begun distributing this extraordinary collection of programs throughout the United States—and around the world. For the background on their development, see our October/November issue.

While we're on the subject of education, we might mention the *EasyLesson* and *EasyQuiz* programs designed for educators. *EasyLesson* lets a teacher create a lesson pool of questions, from which they can then generate tests and quizzes. Teachers have up to seven categories of questions, can print out lessons for clarity, and can restrict a test to questions from a certain category. *EasyQuiz* gives the teacher the ability to administer the tests created by *EasyLesson*. It asks for the student's name and name of the test the student wishes to take, shuffles questions randomly, administers the test in either "flash card" or "multiple choice" mode, and grades the student at the end of the session.

When it comes to keeping track of finances and accounting procedures, the 64 is right there with the *EasyCalc* electronic worksheet, and *EasyFinance* financial analyst. *EasyCalc,* with 65 columns and 999 rows, has the ability to print out all formulas and assumptions, provides selective row reporting and printing, instant "what if" type calculations, and allows up to 20 user-defined functions per spreadsheet. It can be used to prepare budgets, cash flow forecasting, product and resource plans, stock fluctuations, trend analyses—and much more.

EasyFinance can help you decide such financial matters as whether to buy or lease, and provides analyses of loans, paybacks, the future value of ordinary annuities and annuities due, future/present values and profit margin. Using *EasyFinance,* you can find out what your monthly payments will be on a loan, the effects of inflation, how much money is tax deductible, and even how much you can afford to borrow.

After you've worked out all your analyses on *EasyCalc* and *EasyFinance,* prepare high quality charts and graphs to illustrate them using *EasyPlot*. *EasyPlot* features full-page printing of charts and graphs, integration with *EasyCalc*—or the option for the user to input data directly. Not only that, but it can plot up to four data series at a time.

Did we hear someone choke on the phrase "there's not much software for the 64"? Let's go on. Take, for instance, Commodore's *MailMate,* a full-featured name and address program for serious applications. It allows entry, change or deletion of a name and address by number or name, can search any field for creating special lists, and can create a complete printout of the system or generate one or two-abreast labels. It can also interface with the *Word Machine*.

If you've had it with missed appointments, lost time and wasted resources, *EasySchedule* can help. *EasySchedule* lets you tailor certain areas of the program to your specific needs, so you can schedule your time and resources for optimum productivity. If you've ever wished you had a secretary, you'll be glad to know *EasySchedule* can be your appointment diary, your project planner and task coordinator. It shows you the day, week, month and year at a glance, to let you see scheduled items by time slots. The ZOOM option lets you zero in on smaller and smaller time slots. And you can even organize data by priorities.

Anyone who's ever misplaced or misfiled essential information knows how frustrating it can be. That's where *EasyFile* comes in. This database allows the user to define the data, and set up how it looks on the screen. Users can enter, edit and retrieve that data in any manner they choose. And, since the program is memory resident, the entire disk is free for data. That's a lot of information—and, of course, fewer headaches for you.

Finally, you should be seeing the much-anticipated *PET Emulator* for the 64 very soon, if not sooner. The *Emulator* allows many existing PET programs to run on the 64. Since there has been a tremendous amount of software written for the PET, especially in education, the *Emulator* will help make the inevitable transition to the much more sophisticated 64 significantly easier for those who've been devoted PET users.

This is by no means the end of the list. In fact, it's only the tiniest beginning—and that is no exaggeration. If, as many people think, the Commodore 64 will be the industry standard—the computer by which all others are measured—for the next several years, you can be sure there will be an ocean of software before too long. As it is, Commodore itself will be marketing products such as a *Commodore 64 Assembler*, a *BASIC Language Tutorial*, a *Video and Music Support Package* (like the VIC 20's Super Expander), and much more. And that doesn't even take into account the many independent software producers who are already jumping onto the 64 bandwagon. We'll do our best to keep you informed.

Robert Lane

Commodore Takes on New President for North America

Robert H. Lane, formerly a top manager with Northern Telecom in Canada and Europe, has joined Commodore as president of North American operations. Lane's job is to "help minimize growing pains, while maintaining the unique spirit and philosophy that have made Commodore a leader in microcomputing," according to Jack Tramiel, president of Commodore International.

Lane has a broad-based background in international business, finance, marketing and production. Most recently, before joining Commodore, he served as president and chief executive of Northern Telecom's $200 million NEDCO division in Canada and was president of European operations.

His academic credentials include an M.B.A. from the University of Western Ontario and a B.A. in psychology from the University of Toronto. He is a native of Canada, is married and has two children.

Commodore Announces Record Sales

In the third quarter of 1982 Commodore as a whole achieved record earnings of $103 million, exceeding earnings in the same quarter of 1981 by almost $50 million. As Commodore had predicted earlier this year, United States sales took the biggest jump—more than quadrupling from $12 million in 1981 to $57 million in 1982—and for the first time accounted for more than half the company's total earnings.

The company has for some time, now, been the leading computer company in Europe—most notably Britain and Germany—and last year shipped more computers worldwide than any major competitors. This year the company's focus on the U.S. market has paid off, giving Commodore a much larger market share in the States than it held previously, and beefing up total earnings. Before too much longer the company plans to be number one in the United States, as well as Europe, with high-quality products that are affordable to the average household.

NEW!

The DeskTopper

$49.95
plus shipping

Holds the Commodore 64 or VIC-20

A beautiful oak stand that's strong and good looking. The **DeskTopper** hides all of those ugly connector cables. Your computer system will look great at home or at work. Finally, everything fits neatly on a desktop.

The **DeskTopper** holds one or two disk drives, a cassette recorder, a monitor or TV set, and a VIC-20 or 64. There's plenty of room in the back of the **DeskTopper** to hide other computer accessories: power supply, modem, RF modulator, game cartridges, memory expansion, and more.

You can get your **DeskTopper** from your local Commodore dealer, or directly from **Madison Computer.** Just call us at **608-255-5552.** Charge it on your Visa or MasterCard.

The oak **DeskTopper is only $49.95 . . .** a beautiful way to organize your computer system.

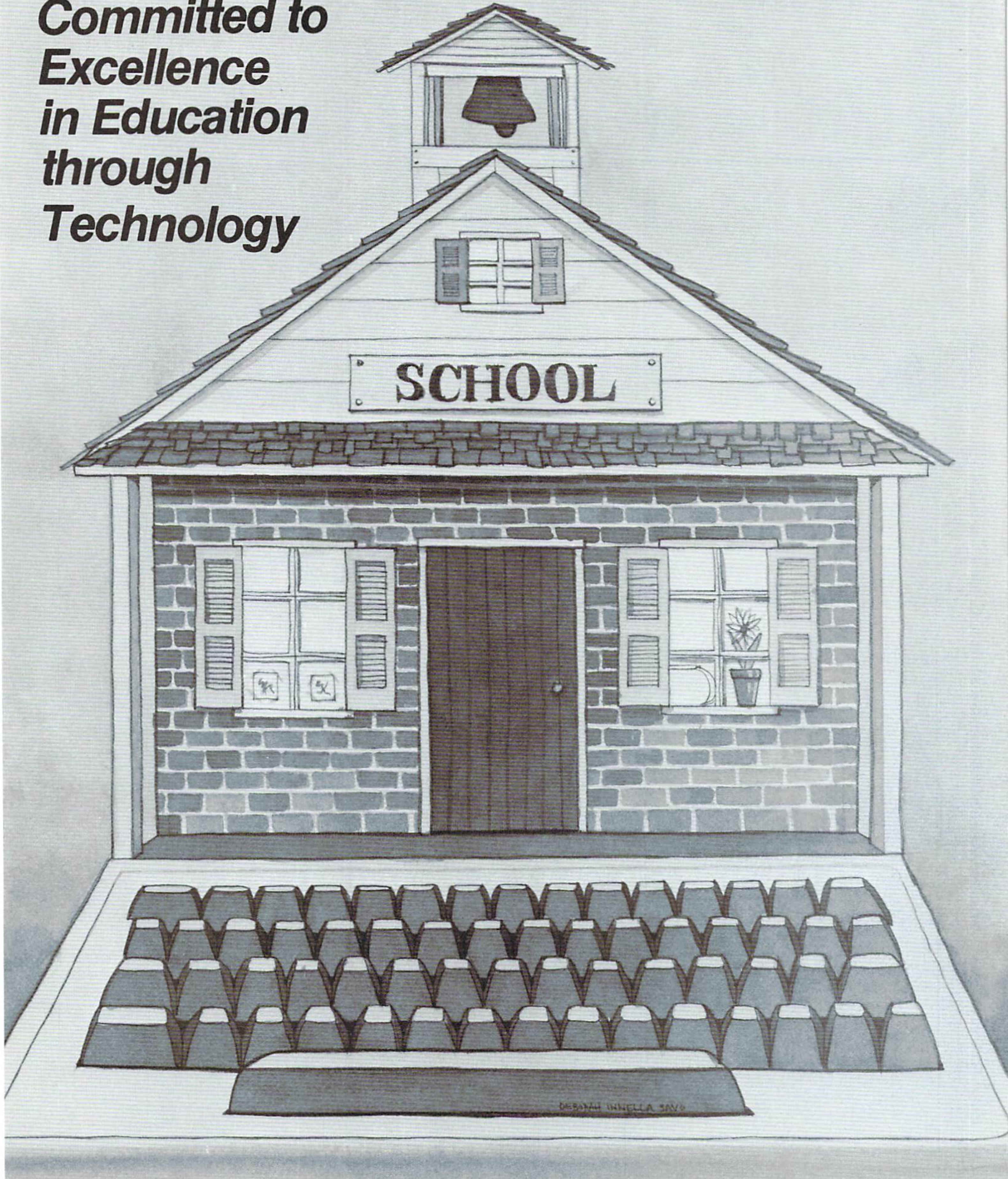
Madison Computer 1825 Monroe, Madison, WI 53711

Call To Order
608-255-5552

Charge it on
Visa or MasterCard
Dealer Inquiries Invited

Circle #10 on the Reader Service Card

Commodore: Committed to Excellence in Education through Technology



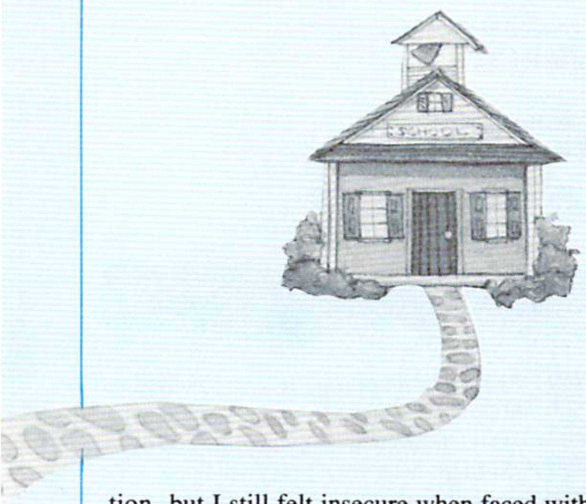
I Wanted to "Go Back to School With a Microcomputer"

by Doris Dickenson

It was surprising enough to learn that I had won a Commodore 64 microcomputer in the *Instructor Magazine* essay contest last spring, but the prospect of having the only computer in our three-school district was overwhelming. Interest in computers and computer education for our students seems to be increasing proportionately to our decreasing budget. Therefore it had been quite easy for me to write the 250-word essay for instructors on "Why I want to go back to school with a microcomputer" that won me the 64.

Even before I received my computer, I began to have concerns about its security. Though theft is not normally a serious concern in our area, the high interest in computers among our student age group could make it a great temptation. I was pleasantly surprised to find that my Commodore 64 is extremely portable. It is no problem to bring it home almost every night to get acquainted with it, and return it easily to the classroom the next day. I also found that it will fit neatly on the shelf of a small cupboard I fitted with a lock when I don't want to take it home. Problem number one solved!

At first I was fearful about my own ability to make use of the computer. Just six months ago I was ready to leave computers to the younger generation of teachers and finish up my five years before retirement without complications. But a computer programmer married into my family, and curiosity got the best of me. I took a two-day computer literacy course through a local college—and got hooked on computers. I also took a three-week class during summer vaca-



tion, but I still felt insecure when faced with the fact of my own computer.

I shouldn't have worried. I found the user's manual accompanying my Commodore 64 was a really understandable and thorough teaching tool and I soon forgot my fears—and settled for sheer enjoyment. Operating the computer is a relaxing and challenging pastime now.

Having the computer in the district has stimulated much interest as well as concern among the teaching staff and has helped force us to face decisions about computer literacy in-service, policies, and fund-raising to buy computers. Our Parent Club has been extremely supportive.

I still had to tackle the problem of what to do with the computer in the classroom. Without any computers, our district has had no computer policy or guidelines and I am completely free to control the use and care of my computer. That suits me just fine.

When the word got around that my classroom had a computer, there was an incredible amount of interest among our fourth and fifth grade students. (Our school is gr. 3-5.) During the summer class we had been warned not to overly extend ourselves and promise more than we could deal with realistically in our computer programs. But it is so hard to turn down a lot of enthusiastic children. I even had one mother who wanted to come after school and learn to use my computer.

This is the solution I worked out. I am teaching computer literacy (history, usage in society, hardware, and general operation) to our two fourth grade classes in weekly sessions by exchanging class times with the other teacher. The computer itself is in the background, and I expect to keep it that way in the group situation for some time.

For individual students, my philosophy is one of making students independent learners, so my goal is to have students working on their own at the computer, while I attend to my duties with the rest of the class.

My solution has been to rewrite the Commodore 64 user's manual in terms that were user friendly, informative, and at a level a strong fourth-grade reader could understand. I am doing it in a series of short, separate booklets—a do-it-yourself manual for children. I might note that we do not currently have any peripherals for our computer, so prepared programs, if they were available, would not be appropriate.

At present, in my room, students whom I have selected as responsible, interested, and capable, are free to use the computer when any regularly assigned work is complete. I limit it to two students working in our Computer Corner at a time, and we usually have some interested observers.

By using my classroom aide, I have freed my schedule for a half-hour weekly, which I spend with six selected fifth grade students. They alternate time on the computer with theory discussions with me. These students have also been issued "Computer Passes" which permit them to come to my room and use the computer whenever they complete their own classroom work and are excused by their teacher. As they complete each booklet of the manual, they will be tested and issued a "Computer License" which permits them to tutor another student at that level. I have been amazed at how easily they have adapted to using the computer and how they are already beginning to experiment on their own.

One further project I anticipate in the near future is to add the computer to my classroom reward system. As a result of the upcoming aluminum can drive, I plan to purchase joy sticks and some game cartridges for our Commodore 64. The children will feel involved in the purchase if I use the money they earn. I plan to make ten minutes of "game time" on the computer one of the rewards they may "purchase" with the stars they earn as part of our classroom reward system. This should really be a strong motivator.

At present I have no plans for purchasing software for my computer. I have not had an opportunity to evaluate what is currently available for the 64, but I am sure as more becomes available, I will look for unique programs that will add something special to my curriculum. I am *really* looking forward to writing or adapting a program to help with my extensive student records. In addition, my co-worker, a music teacher and "non-computer", is anxious to experiment with the sound and music synthesizer. So, at this point, I feel we have only scratched the surface of the potential of our microcomputer in the classroom.

A footnote to this article is that one of the other teachers in the school just earned enough money in a walk-a-thon/jog-a-thon to purchase the affordable Commodore 64 for his own classroom. I am really excited at the prospect of having someone else in the school to share ideas and materials with.

Public Domain Software for Commodore Computers

by
Neil Harris

Dr. Senese, Assistant Secretary of the U.S. Department of Education (left) with Jack Tramiel, founder and president of Commodore International (center) with children of the United Nations International School.

I don't know about you, but I get annoyed when I see some commercials for other computers. Like the one where the spokesman points to a VIC 20 and says, "not much software". That's been a rap on our computers since the early days, and it's just not true. Our commercial software is out in force, and stands up (on the whole) better than theirs. But there are other sources of software, and we really overwhelm them there.

Once you're a computer owner for any length of time, you realize that most of your software collection isn't bought, it accumulates. It consists mostly of little programs passed from user to user.

These programs aren't copyrighted like commercial programs. They also aren't slick like commercial programs, and even might not work as well as commercial programs. They do have a big advantage, because they're free.

Legally, these programs belong to the public domain. Astute collectors of public domain programs can obtain huge libraries of software. There are even semi-public repositories of programs, who'll make copies for you at low cost.

The biggest collection of programs I ever heard of belongs to the Toronto PET User Group. These people have collected thousands of programs, from games to education to mathematics to utilities, and made them available to members.

Enter Commodore. It was decided to make these—and many other—programs more widely available. The main problem was that many of these programs weren't in such good shape. In conjunction with the Ontario Department of Education and the government of Ontario, Commodore hired students on their summer vacation, who were paid to translate the programs to a common format. Under the direction of Frank Winter in Commodore's Toronto office, they used standard subroutines and followed strict guidelines. Color and sound were added using the Commodore 64 computer. These programs were placed in the public domain, and distributed throughout Canada this past September.

Commodore held a press conference at the St. Moritz Hotel in New York City. Invited were Sylvia Fuhrman, special representative of the Secretary General of the United Nations, and Dr. Donald J. Senese, of the U.S. Department of Education. Sylvia Fuhrman brought along four students from the U.N. International School to demonstrate the programs. Speaking for Commodore were Jack Tramiel, president and founder, Kit Spencer, vice president marketing, and Frank Winter. At this time, Commodore officially released the translated programs to the public domain, worldwide. Anyone is free to copy and distribute these programs. Dealers can purchase copies of the complete set from Commodore for just the cost of the diskettes and copying, and then distribute them to schools and other customers.

Next summer, Commodore expects to translate another 1000 programs to common standard and release them. Commodore 64 users: watch for a batch of top quality programs, available for free.

VIC Long Division

This little program, converted by Rick Cotton from the original by Edward M. Lichten, sets up and performs long division problems on your screen. Set up the problem, then type in your answer, one digit at a time. The program will do all the multiplication and subtraction operations for each digit you supply in the answer.

```
1 REM LONG DIVISION, BY EDWARD M LICHTEN, CONVERTED BY *RICK COTTON*
10 DEF FNT(T)=15-INT(LOG(T)/LOG(10))
20 DIM Q$(4),Q(4),B(3)
30 PRINT"TEACHING LONG DIVISION":PRINT
40 PRINT"  THE VIC WILL TEACH  YOU LONG DIVISION  BY";
45 PRINT" BREAKING EVERY PROBLEM  INTO  LITTLE";
50 PRINT" ONES.  FOLLOW  THE  PROBLEM THROUGH ONE STEP  AT A TIME.  WHEN YOU GE
T  GOOD ";
60 PRINT"THERE WILL BE NO HELP! TO STOP TYPE END"
70 PRINT:INPUT"TYPE LEVEL(1-7)";G
89 REM L IS LIMIT OF DIVISOR
90 L=G*2:IF G>6THEN L=99
95 PRINT"      FIRST PROBLEM"
100 A=INT(L*RND(TI)+1):BB=INT(998*RND(TI)+1)
120 IFA=1 OR BB<ATHEN 100
130 D=A*BB
135 IFD<1000THEN100
139 REM B(1) IS HUNDREDTH'S PLACE;B(2);TENS;B(3);ONE'S
140 B(1)=INT(BB/100):B(2)=INT((BB-100*B(1))/10)
150 B(3)=BB-100*B(1)-10*B(2)
200 PRINT"";
210 PRINTTAB(10);" ┌──────"
220 PRINTTAB(6);A;TAB(10);" |";TAB(FNT(D));D;:PRINT""
250 FORI=1TO3
260 IFSW=1THEN Q$(I)=STR$(B(I)):PRINTTAB(FNT(BB));BB:GOTO330
299 GET ANSWER FROM STUDENT
300 IF I=1 AND B(1)=0THEN Q$(1)="0":GOTO 320
305 GET Q$(I):IFQ$(I)=""THEN305
310 IFQ$(I)="E"THEN999
320 PRINTTAB(13+I);Q$(I)
330 Q(I)=VAL(Q$(I))
335 D=D-A*Q(I)*INT(1000/10↑I+.05)
338 IF NC=1THEN PRINT"":GOTO410
339 REM DETERMINE LINE TO PRINT REMAINDER
340 IF I=1THENPRINT""
345 IFI=2THENPRINT""
350 IFI=3THENPRINT""
360 IF Q(I)=0THEN PRINTTAB(15);0:GOTO380
370 PRINTTAB(FNT(A*Q(I))-3+I);-A*Q(I)*1000/10↑I
380 PRINTTAB(12);"-----"
400 IFD=0THEN PRINTTAB(15);0:GOTO407
```

```
402 IF D<0 THEN D=-D:GOTO420
405 PRINTTAB(FNT(D));D
407 IFI=1THENPRINT"TTTTTTT"
409 IFI=2THENPRINT"TTTTTTTTTT"
410 NEXTI:FORJ=1TO999:NEXTJ
413 PRINT"XXXX"
415 IFSW=1THEN FORZ=1TO5000:NEXTZ:PRINT"J  LET'S TRY ANOTHERXX":GOTO445
420 IF D>0ANDSW<>2THEN PRINT"J  WRONG-LET'S REVIEWXX":SW=1:GOTO130
430 IF D=0THEN C=C+1:PRINT"J CORRECT! VERY GOOD!!XX"
435 IFSW=2ANDD<>0THENPRINT"J  WRONG-TRY AGAINXX":GOTO130
440 IF C>10 THEN700
445 IFSW=1THENSW=0
460 GOTO100
700 IFFT=0THENPRINT"JX   IF YOU DO NOT WANT COMPUTER H LP TYPE Y":INPUTY$:IFY$
="Y"THENSW=2
710 FT=1
720 GOTO100
999 PRINT"JX   YOU  GOT ";C;" LONG    DIVISION PROBLEMS"
1000 PRINT" CORRECT! VERY GOOD!!       XBYE NOW!"
1010 END

READY.
```

Commodore in the Schools

by Terry Anders

This is an excerpt from the monthly newsletter published by the education marketing department of Commodore's midwest regional office, 2246 N. Palmer Drive, Schaumburg, IL 60195.

New City, a small, private elementary school in St. Louis serves students from age three through 12. They have a computer lab of 8 PETs networked with the headmaster networking system linked to an 8050 drive and 8023 printer. Donna White, a teacher and computer specialist for the school brings her 64 and uses it in the classrooms!

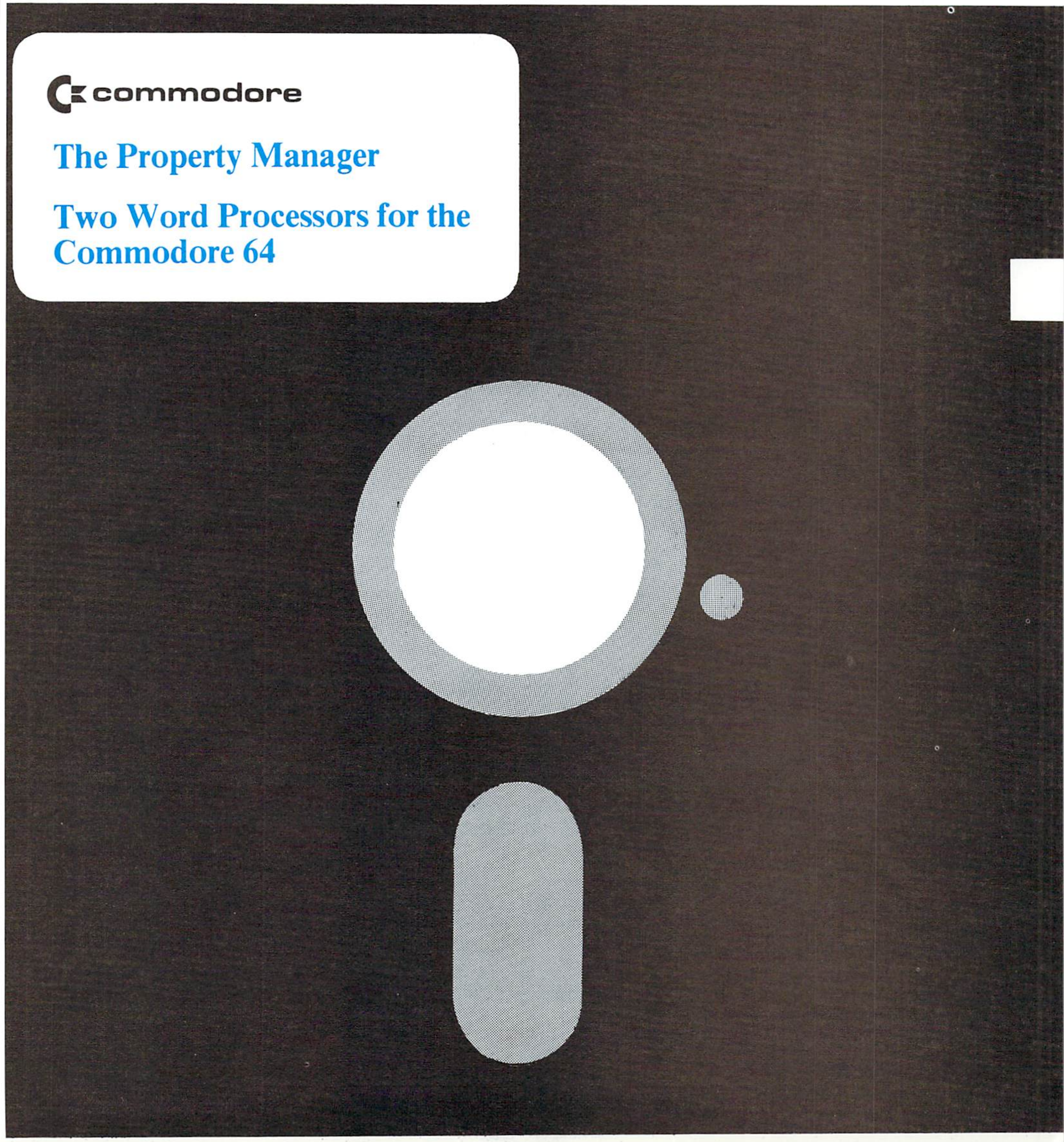
New City is committed to the fullest use of their computers. The lab is used extensively throughout the day for computer assisted instruction using public domain software selected by the teachers as well as for computer literacy training by classroom teachers. Students have responded enthusiastically to the program and enjoy the electronic mail facility of the headmaster!

The computers are being used throughout the curriculum at New City. A curriculum is being written so that students will have thorough training in the applications of computers as well as broad experience using them. Presently, the physical education teacher is using a computer to compile, store, evaluate and print the results of physical fitness tests; the librarian is putting the card catalogue on disk as well as cataloging and documenting the public domain software; the art teacher is excited about the graphics capabilities of the 64; the French teacher is awaiting the voice synthesizer and the music teacher anticipates multiple use of the synthesizer on the 64!

With uses such as these and ideas growing daily, New City needs more computers! To address their funding needs, they began a series of afternoon and evening classes. These classes teach children and adults at all levels. The program is full and the demand high. Most importantly, these classes will generate enough revenue to allow the purchase of 64's for each classroom and some expansion of their PET lab. New City expects its program (and revenue) to grow! !

New City School recognized the need to address the issue of computer literacy training for its students. By allowing involvement on the part of all teachers and being creative in the search for the monies to buy more hardware, they are insuring their students a functional role in a technological age.

BUSINESS


commodore
The Property Manager
Two Word Processors for the Commodore 64

The Property Manager

Property Management System from National Micro Business Systems

reviewed by Diane Le Bold

One of the most tedious and time-consuming parts of managing any kind of rental property is in keeping track of income and disbursements—and producing regular income statements. For managers of several large multi-unit properties, these tasks quickly become monumental and liable to error if they are done by hand. Those who have been relying on pencil and paper up until now will be glad to know that the *Property Manager* system for the CBM 8032 can ease the burden of complex record-keeping, and produce regular printed reports as needed for each property with a minimum of clerical effort.

The Property Manager allows you to keep track of a virtually unlimited number of properties, with up to 150 units per property. If a particular property has more than 150 units, you simply divide the property into two entries with slightly different code names. It allows entry of up to 175 cash disbursements per property per rent cycle, and can list these disbursements by check number, type of disbursement or payee.

At the end of each rent cycle, the Property Manager can call up a list of delinquent lessees, or tell you whose leases are expiring, so those people can be sent appropriate notification. Should you for any reason need an alphabetic list of lessees for a property, the Property Manager will produce one for you.

The system will also list income statements for each property, automatically update all unit files at the end of each rent cycle and produce complete financial reports on all properties. It allows you to record cash receipts from things like laundry and vending machines, and generally takes into account most of the records that property managers need to keep.

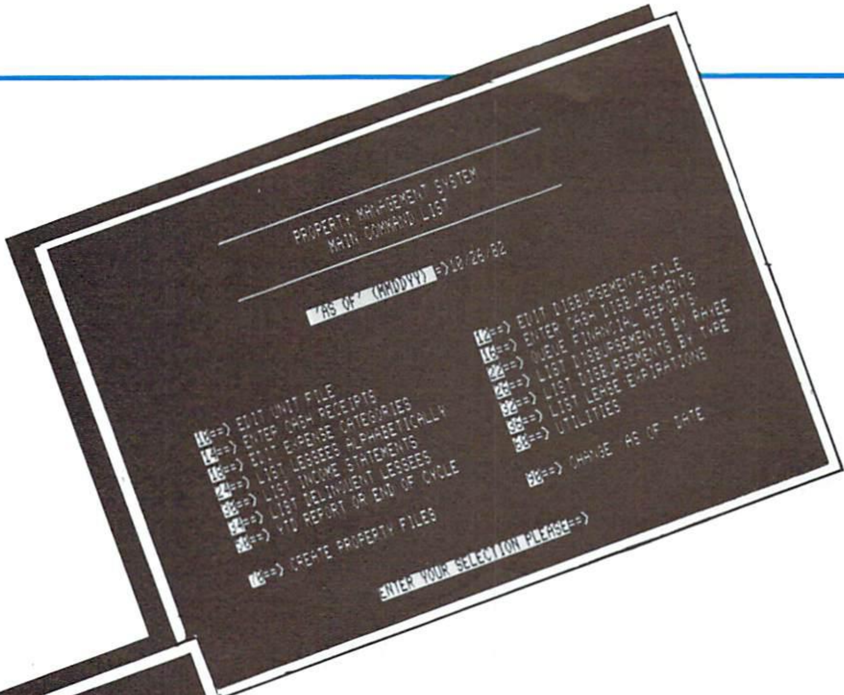
I also might point out that the "property" in question does not necessarily have to be real estate. The system would be equally effective for any type of renting or leasing operation — say, for instance a car rental company, in which each "unit" is a vehicle, rather than a living space.

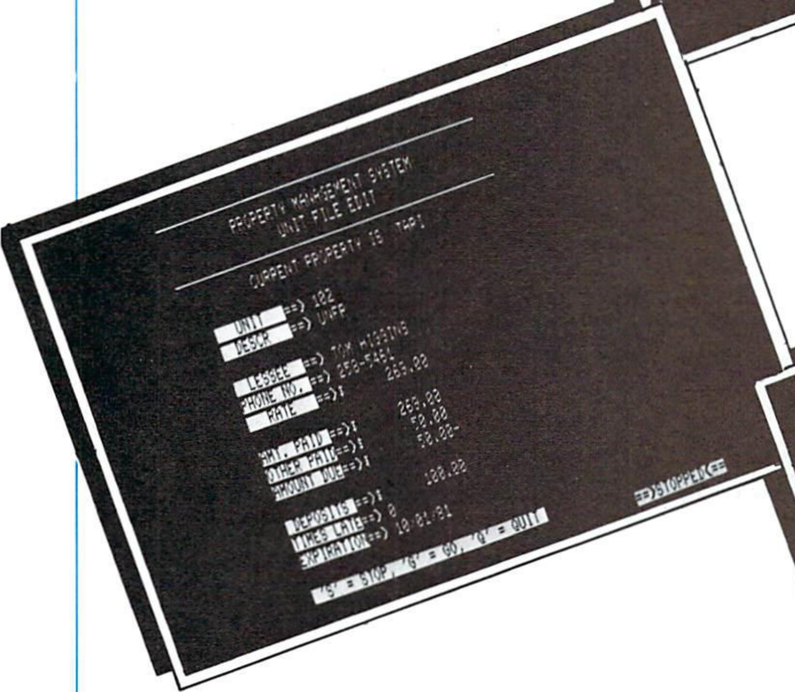
The system is designed for non-computerists, and comes with a complete, clearly written manual that explains all its functions and commands, and provides an overview of the system's capabilities. The user begins by creating a file for each property. This file contains the property's code name (up to six characters), full name, address, number of units and beginning bank balance. It sets up the number of checks allowed to be written during each rent cycle, and establishes codes for each expense category.

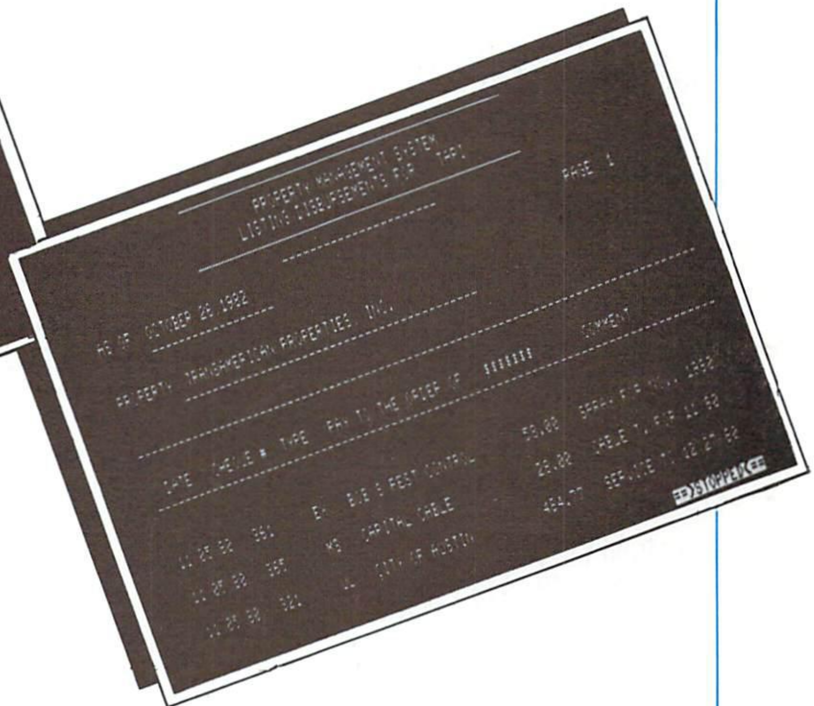
The main weakness in this part of the system, from a user's point of view, is that once a property file is established, the information in it is virtually written in stone, since there is no way to edit most of it. Therefore, should the post office change any part of the property's address, for instance—certainly not unheard of—you have no way of going into the property file and making the change there. And although you can change the names of your expense categories, you cannot change the codes themselves, once they are on file. (You also have no access to what the codes are as you go about entering them in the cash disbursement file, so you'd best keep a list somewhere for your reference.)

Should you discover that you need to write more checks each month than you originally thought, you have no way of editing that part of the property file, either. That means, if you've allotted 50 checks per cycle to a property and need to write 52, the system won't let you do it without closing that month's books, first. The obvious solution is to cover yourself by allowing the full 175 checks per cycle for each property—but if you have to do that for every entry, what's the purpose of even having a choice?

Another seemingly small but potentially clumsy characteristic of the system is that it allows individual units within a property to be designated only by numbers. This could become a pain in the neck for large properties that have, for instance, several buildings and use both letters and num-







bers to differentiate among units. I've lived in several complexes like that, where the only way to differentiate among the several "apartment 102's", for example, was to add the code for the building—usually a letter.

To avoid problems in setting up your main property files, then, be sure you have all your categories and facts well thought out before you enter anything. The manual suggests this at the outset, although it doesn't explain in detail why it's so important to do so.

It's also important to remember—and the manual warns you of this in large type—that each time you call up the list of delinquent lessees, the "times late" section of their individual unit file is added to. That means if Tom Smith is late on his rent this month, and has been late twice before (a fact recorded on his individual unit file), and you call up the list of who's delinquent, Tom's individual file will change to say he's been delinquent three times—which is correct. However, if you feel like sneaking another look the next day, Tom's file will then say he's been late four times, which is not correct. The advantages of having an automatic update on every delinquent lessee's file would seem to outweigh this little quirk, though, so you just have to remember to be discriminating.

In spite of these several idiosyncrasies, the Property Manager seems to be, in general, a valuable tool for keeping fairly complex records. Its automatic cross-referencing and updating capabilities undoubtedly increase the ease and accuracy of record keeping for managers of leased and rented properties—whether those properties are in real estate or some other form, such as cars and trucks.

The latest software for your COMMODORE 64! The Word Machine is available through most dealers at press time,and Easy Script is in final testing, and will be out early in 1983.

Two Word Processors for the Commodore 64

reviewed by Neil Harris

Word processing is one of the single most useful functions for a computer. An editor once compared the difference between a typewriter and a CBM word processor to the difference between chiselling words in stone and molding them from clay. Personally, I'm a two-finger typist, and my production on a plain typewriter is only about 4 pages per hour. On a word processor, I can triple that without any of the aggravation that results from "stupid" typewriters.

What do you do with a word processor? Anything from letters to manuscripts to magazines. I once published an amateur magazine for a local science fiction club, typesetting it with a word processor. I could create neat, justified columns of text. It was a simple matter to take the columns and paste them up into a good-looking format.

EasyScript, for the Commodore 64, gives you some extremely sophisticated capabilities for word processing. First, this is a 100% machine language program, offering flexibility and speed that BASIC programs just can't match. It lets you use letter-quality printers, like the NEC Spinwriter and Diablo printers, if you need the extra quality they offer for manuscripts and business letters. It also allows use with tape for transporting documents.

For those of you who like to see on the screen what you'll see on the page, EasyScript lets you adjust the screen width to any size from 20 to 240 characters. When using widths larger than the 64's 40 columns, the program scrolls *sideways* like the carriage of a typewriter. Personally, I just like to use 40 columns, because you can see all the text you're working with.

I am very impressed with the advanced capabilities of this program. I'm used to working with WordPro on the CBM 8032, and EasyScript can do anything WordPro can do, and then some. For example, WordPro allows you to create heading lines at the top of each page, which is essential in manuscripts. EasyScript lets you adjust the margins of the headings independently of the page margins, something I've often wanted to do but couldn't with WordPro.

EasyScript makes it simple to work with the disk while using text, another feature that WordPro doesn't have. You can read the disk directory in EasyScript without disturbing your text.

Some of EasyScript's features include moving and storing any section of the text (any characters, not just whole lines), moving the cursor by lines and pages (slowly or quickly), filling in blocks of text from another text file (for personalized form letters), deleting any section of text, hunting for words (you can choose to automatically replace them with other words), decimal mode for lining up columns of numbers, horizontal and vertical tabs that can be stored with the text, and more. Printer-oriented commands let you use bold face, underlining, super- and sub-script, programmable characters, and other features for special printers.

EasyScript is truly a professional word processing system, one that allows for any possible need. It works with almost any printer that can be hooked up to the Commodore 64, and can create an astonishing assortment of document appearances. Compared to word processors on much more expensive computers, it stands up well, and for $100, it is a good investment.

The Word Machine comes on a diskette for use with the Commodore 64 computer and 1541 disk drive. It is a BASIC program, which limits its capabilities, at least when compared with EasyScript. For $29.95, including the mailing list software called **The Name Machine**, it is a bargain.

The program begins by allowing you to change the colors of the screen, border, and characters—helpful for fuzzy home TV sets. I prefer to work with black letters on a white page, like a typewriter, but you can adust for your own idiosyncracies.

It then asks you if you're working on tape, disk, or both. Since the program comes on a diskette, I can't imagine too many uses for storing on tape. It is helpful if you're mailing a copy of your text to someone, since cassettes tend to travel with less loss of data than diskettes.

You also type in today's date, which is automatically tacked onto the beginning of the text on disk or tape. This lets you keep track of the latest revisions.

Next, the program displays the main menu. This menu lets you begin typing text, recall text from storage, edit an existing document, print out your text, store your text, display the text in printed form, search for words and replace them with other words, and process form letters (using the Name Machine's mailing list).

Creation of a document is very painless. Even though the program is in BASIC, it has no problem keeping up with very quick typing. It won't let you fix mistakes in typing at this time; you must wait and go into edit mode later on for this.

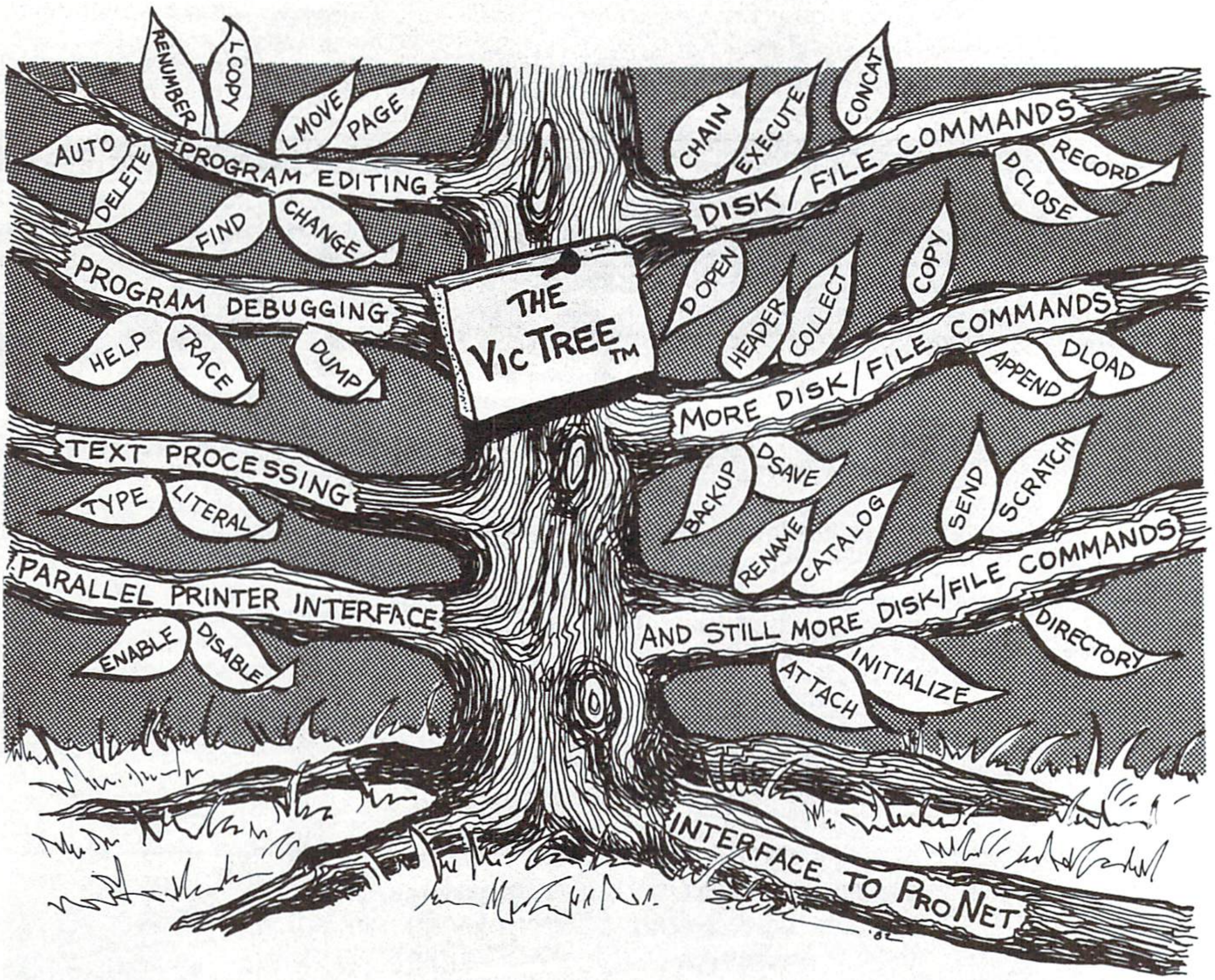
Once created, text should be stored—just in case. You wouldn't want to lose all your work due to a power failure.

Now you can go back and change your text using the Edit mode. This lets you select a spot in your text to change, or add more text in the middle. Here you must be careful to type slowly, for the program has quite a bit of work to do. The BASIC program won't keep up with any but the slowest typing rates when editing your text.

The printout section of the program lets you adjust several parameters. You can single or double space. You can set the number of characters per line, from 20 to 80 characters. The page size is adjustable, so customized forms can be utilized with this program.

The Word Machine is a simple word processor for the beginner who just wants to create text and print it out on a printer like the 1525 or 1520. It requires patience to create, edit, and format a document to your liking—but on the other hand, it costs only $29.95 on disk, including **The Name Machine**, and is a good introduction for those with simple word processing needs. C=

Skyles Electric Works Presents



The VicTree™

...Leaves your new Vic (or CBM 64) with 35 additional commands.
...Branches out to most BASIC 4.0 programs.
...Roots into most printers.

New from Skyles: the VicTree, a coordinated hardware and software package that allows your Vic to branch out in unbelievable directions and makes it easier than ever to do BASIC programming, debugging and to access your disk. And the new VicTree provides routines to interface the Vic to the powerful ProNet local network. 8kb of ROM — 4kb for the BASIC commands, 4kb for disk commands and interfacing to ProNet — plus 4kb of RAM for miscellaneous storage. Perfect not only for the new Vic but also for the Commodore 64. Unbelievably simple to use and to install, the VicTree gives you all the additional BASIC 4.0 commands to allow most BASIC 4.0 programs to work on your new Vic or CBM 64.

Now only $89.95...or **$99.95** complete with Centronics standard printer cable. (Cable alone **$19.95.**) Available now from your local dealer or order through your Visa or MasterCard toll free: (800) 227-9998 (California, Canada, Alaska, Hawaii: (415) 965-1735) or send check or money order directly to:

Skyles Electric Works

231E South Whisman Road
Mountain View, CA 94041
(415) 965-1735

Circle #12 on the Reader Service Card

Bruce Cameron, WA4UZM
and David Cameron, Ex-WA4VQR

Seeing RTTY on the VIC

If you own a VIC 20 computer and would like to look at RTTY, here is the proverbial "quick and dirty" method. We have reduced both hardware and programs to a minimum, but in a form that can be elaborated later if you wish.

The first requirement is a stable CW receiver, with a sharp filter. We used a Collins 52S1 but any good one will do. We tune only one of the RTTY frequencies, the one called "space." More sophisticated demodulators look at both mark and space, but all the information we need is in "space", although without protection from nosie or fading. We take the audio output, couple it through a matching transformer, rectify it, smooth it with a capacitor, and feed the result to the base of a switching transistor. This in turn tells the computer "0" or "1" and the program does the rest. The diagram on the next page tells all except how much work it took to figure it out.

Since the VIC 20 is an ASCII machine, that is the easiest thing to take off the air. W1AW sends their bulletins first in Baudot and then repeats them in 110 Baud ASCII. A short program in BASIC is required. (See below) Note the number "3" in line 10. This controls the Baud rate. 1 is 50, 2 is 75, 3 is 110, 4 is 134.5, 5 is 150, 6 is 300 and so on. Other details can be found in the *VIC 20 Programmer's Reference Guide,* from Howard Sams and Company, Indianapolis.

Now this is well and good if you are copying modern stations, but most hams and commercial stations still use the older Baudot, or Murray, code. This is based on a typewriter with two shifts to select either of two characters for each key, one called "ltrs" and the other called "figs" and the shifts automatically lock until you hit the other ones. Some non-printing characters include line feed, carriage return; some ring bells, and so on.

The locked-shift characters of TTY open up some nasty problems. The character that tells the machine to do "ltrs" is all mark. This means that if you miss any one of the five, the machine will stay in figures until it sees the next "ltrs" shift. For this reason, anytime you send "fig" (so that you can print a punctuation mark or a number) you invite the thing to stay out of letters mode due to a slight bit of noise or fade or whatever. For this reason, since most transmissions are text, it has long been RTTY practice commercially to send one or two carriage returns (to insure getting one), line feed, and ltrs at the beginning of each new line. Even so, you may get a whole line of unwanted figures. Old newspapermen simply learned to read off the letter equivalents of the printed figures.

We can handle that problem in one of two ways. If we use a program that only prints letters, and make "figs" a non-printing space, the machine will print all the text and when it comes to a number it will leave a blank space, print the letter equivalent, leave another space and go on with the text. Thus, WA4UZM will come out WA R UZM.

This provides a very short program, seen on the next page.

Notice that we supply a conversion table and the VIC thereby translates the Baudot signals into ASCII, according to their ordinal number. Pay close attention to the spaces because they are important. They indicate the non-printing characters in their order. Note the figure "1" in the open statement. This means 50 Baud or 67 w.p.m. Baudot. Many commercial stations use this rate, but most hams use 60 w.p.m., or 45 Baud. It might be possible to generate this in the VIC but the programming would be rather complex. An easier way is to do a little surgery. The crystal that is the BASIC clock is on 14.318 Mhz. If we substitute a crystal that is 9/10 of this we slow everything down proportionally and we can get 45 Baud. The crystal is on the front of the board and there is room to mount another and a slide switch just under the front apron. The easy way to make a hole in the plastic case is with a soldering gun and a

sharp knife to clean up the debris. The nominal frequency for the crystal is 12.886 (series resonant) but we used one we had at 12.970. The error does not seem to matter.

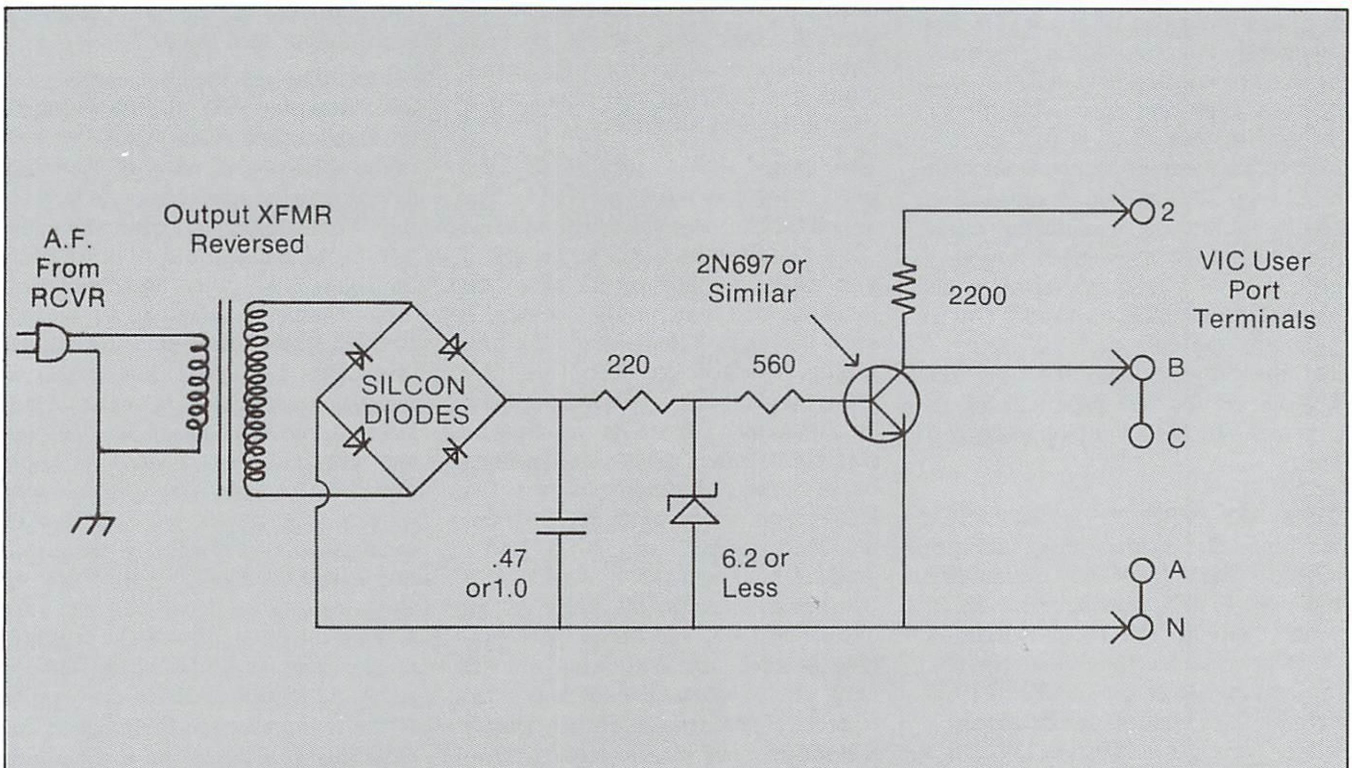
WARNING: The Editors take NO responsibility for elective surgery on VICs. The above procedure will VOID the warranty, and should only be performed by experienced technicians.

If you want a full Baudot program so that you don't have to translate the letter representations of the figures, we have one on page 35.

This program simulated most of the functions of a mechanical TTY, such as a Model 15. It is not set up to transmit, in the interest of simplicity. If you want to transmit, it can be modified, and the VIC S-out connections will trigger the transmitter. On the other hand, if you type as poorly as we do perhaps you would prefer just to print and not let the world share your shame. We like to copy press and to nose around between the ham bands. The selection of two crystals now provides Baud rates of 45, 50, 67, 75, 99, 110, 121, 135, 150, and so on. About the only useful one we do not get is 57 baud (75 w.p.m. Baudot.) Don't try to change crystal frequencies while the VIC is running. (It confuses it completely! Decide which crystal you want and then power up. And don't expect to make sense out of every signal you find on the air. With all the RTTY gear we have ever had we found signals we could not understand. Perhaps the authors did not mean for you to know what they were saying!

We tried any number of gimmicks with this program, such as buffering and dumping print one line at a time. We tried it both with and without carriage returns, (because our lines do not match those coming in.) The only one we think essential, and have kept here, is a way to force the machine back into "ltrs" when it gets stuck incorrectly in "figs." All you need to do is punch "L" on the keyboard and it will print letters again. If you type "F", it will print figures. With bad signals you might have to stand over it, and you may wish to use the Letters Only program instead. We store both on the tape and load whichever one we prefer at the time.

This is probably the simplest and cheapest all-electronic RTTY you can find, and you can move up to something better with no waste, by building a better demodulator, and modifying the programs for transmission and various gimmicks, However, while this project began as an exercise in simple design it has worked so well that we may not get around soon to changing it. Happy snooping!



ASCII Program

```
5 REM "ASCII"
10 OPEN2,2,0,CHR$(163)+CHR$(160)
20 GET#2,C$
30 IFC$>" "ANDC$<"↑"ORC$=CHR$(13)THENPRINTC$;
40 GOTO20

READY.
```

Baudot Letters Only

```
5 REM "BAUDOT LTRS"
10 OPEN2,2,0,CHR$(97)+CHR$(0)
20 T$="E A SIO"+CHR$(13)+"DRJNFCKTZWHYPQOBG MXV "
30 GET#2,C$:IFC$=""THEN30
40 T=ASC(C$)
50 IFT>0THENPRINTMID$(T$,T,1);
60 GOTO30

READY.
```

Full Baudot

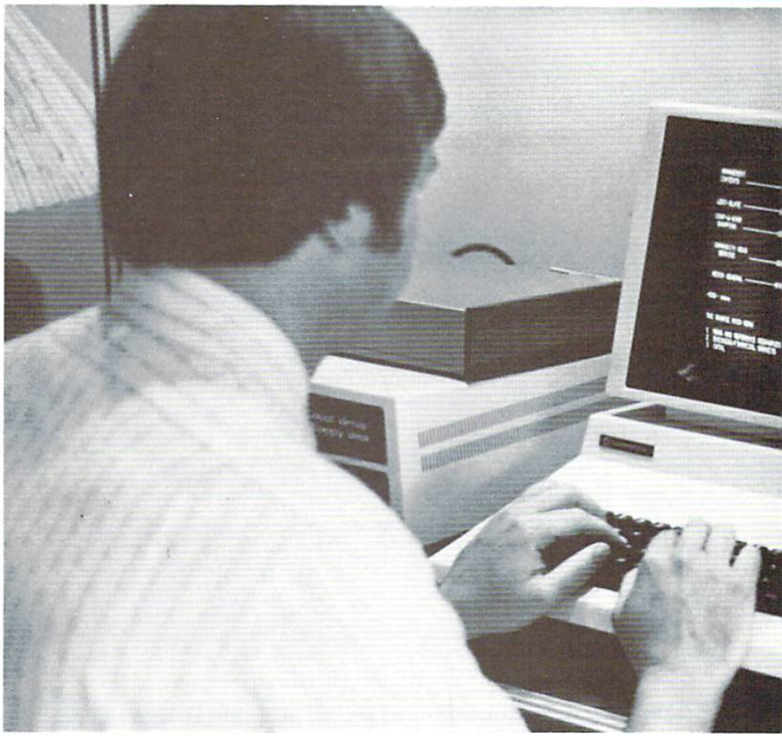
```
5 REM "FULL BAUDOT"
10 OPEN2,2,0,CHR$(97)+CHR$(0)
20 LS=-1
30 LF$=CHR$(10)
40 CR$=CHR$(13)
50 L$="E"+LF$+"A SIO"+CR$+"DRJNFCKTZLWHYPQOBG*MXV*"
60 F$="3"+LF$+"- ´87"+CR$+"$4´,!:(5´)2#60197&*./:*"
100 GET#2,C$:IFC$=""THEN160
110 C=ASC(C$):IFC<1ORC>31THEN100
120 IFLSTHENC$=MID$(L$,C,1)
130 IFNOTLSTHENC$=MID$(F$,C,1)
140 IFC$<>"*"THENPRINTC$;:GOTO160
150 LS=(C=31)
160 GETX$:IFX$=""THEN100
170 IFX$="L"THENLS=-1
175 IFX$="F"THENLS=0
180 GOTO100
```

De-Mystifying BAUDOT to ASCII Conversion Tables

Readers who roll their own RTTY hardware and programs may have been puzzled to see apparently conflicting lists of characters derived from the binary numbers of the Baudot code. An explanation is as follows:

If you look at a punched tape from a Baudot machine you see five places which may be solid or may have a hole. Conventions have been argued concerning what is the most significant and what is the least significant bit. Moreover, you could count spaces (no hole) or marks (holes), and either could be assigned some arbitrary number. Usually the numbers are assigned 1, 2, 4, 8, 16. However, they can just as well be assigned 16, 8, 4, 2, 1. This makes four possible sets of numbers, each giving a discrete (but different) total for each letter and other character. Now the one we use here starts E Line Feed A S I U etc. But a recent article in 73 used a set that starts K Q U # J etc. You could still use the binary representation method with other less rational systems, such as assigning marks the numbers 16, 1, 8, 2, 4. This and other systems will give about equal results, but might confuse the uninitiated even further.

For people who think the world is real and not merely nominal, this may be unsettling, but for others who merely want peace of mind the answer to the question "But who is right?" is either "All of them" or "None of them." Take your pick. C=



COMPUTERIZED HOUSE THAT BUYS & SELLS ELECTRICITY

By Peggy O'Neal

John and Becky Fox both work all day, but while they are away their house bustles with mechanical activity. The dishes are washed and the clothes are dried. The house is kept at a moderate temperature, and the air conditioning automatically starts cooling the rooms abut half an hour before John and Becky are expected home. Fire- and burglar-alarm systems that automatically signal authorities in an emergency protect the premises and when it becomes hot outside, shades roll down to cover windows exposed to the sun.

When Becky and John arrive home and step inside the airlock entry, if the house feels a bit too cool, Becky walks over to the thermostat and punches in a digital code that signals the air-conditioning system to quit working so hard.

John heads for the family room, as he does everyday, on the house's lower level and strikes a series of keys on the home computer, which has been working all day as well. It tells him that more energy was produced by his house today than was used. In fact, enough excess energy was produced that the computer decided to sell some back to the local utility company. A few more keystrokes take him to the accounting ledger and, sure enough, the utility company owes *him*.

These are but a few of the things the computer helps Becky and John Fox with, or will help homeowners like them with, when an experimental home in Atlanta is sold and a family moves in this fall. The home began a six-month open house in April.

It is a house that can produce more energy than it uses and reduce its energy consumption by more than 50 percent over conventional homes by using all of its produced and purchased energy in the most effective way.

At the hub of it all is a Commodore home microcomputer, just like those in many homes across the country.

Called Future I and built by Georgia Power Co. for purchase by a private owner, it is not a futuristic-looking home, except for the 16×32-foot array of photovoltaic-cell solar collectors on its south-facing roof. Actually, the front of the house faces north and as you drive up to it, it looks like any other in its upper-middle-class subdivision.

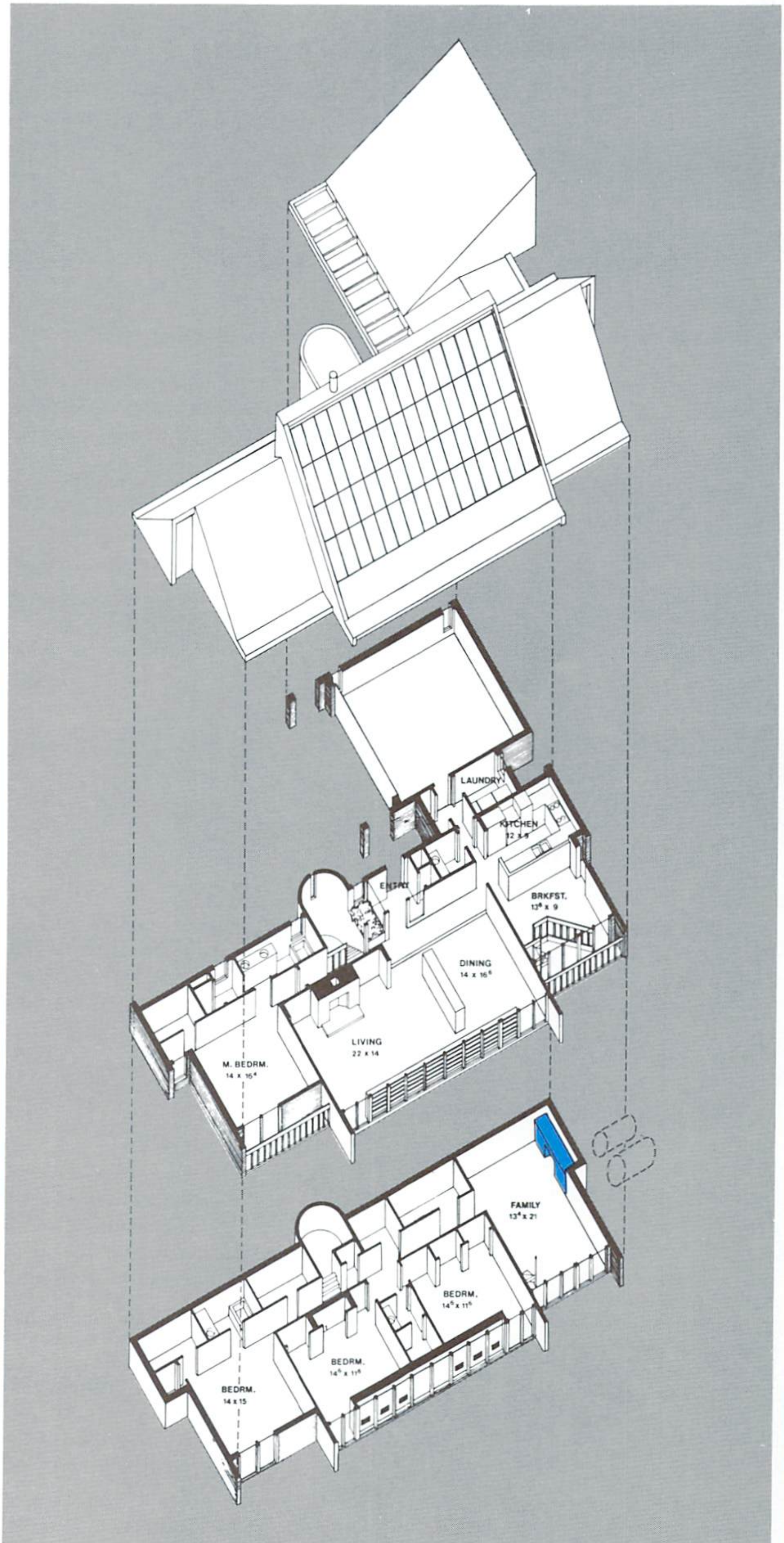
"We did not build a house that's all gadgets. It was intended to be a house like any other," says Gary Birdwell, manager of the project. The architect, Richard Sibly, echoes: "It's a passive-solar house that people can accept."

It is a home that most people would not attempt to design or build, however, for it makes extensive use of a variety of passive-solar features which are expected to provide 63 percent of the home's energy needs. Indeed, many people could not afford to build it, for the unique array of photovoltaics on the roof costs about $120,000 to design, build and install.

But the new homeowner won't have to pay for this most-expensive part of the house. In exchange for the ability to monitor the home's performance for five years, Georgia Power is giving the collectors to the new owners. Officials of the company feel that the gift will more than pay for itself in recommendations the company can make to power users, thereby forestalling or eliminating the need to build more generating stations.

During the six-month open-house period, Bill Cooper, an energetic computer programmer with experience in industrial applications, is refining and expanding its computer operations.

So far, the Commodore CBM 8032 (an updated model of the PET) in the family room has been programmed for its major function: load management of the energy produced by the photovoltaics. The program, says Bill



Cooper, is based on the time-of-day rate structure of the utility company.

During the day, when electricity is expensive to buy, the photovoltaics heat or cool the home, using a heat-pump heating and cooling system. But the photovoltaics are expected to produce about 4 kilowatts of electricity, and on moderate days not all of this energy will be needed to cool or heat the home.

This is where the computer comes into play, for since the homeowner won't be paid as much for power sold back to the utility as he'll pay for power purchased from it, the most efficient use is to store it somewhere in the home.

"The main idea is to store excess energy somewhere," says Birdwell. It may be stored in the hot-water heater, so the computer routes the unused energy to that use. If there is still no need for heating or cooling the house, another possible storage use is to recharge the cold-water storage-tank system which provides a secondary cooling system for the house. The computer must know whether it is winter or summer, though, because there is no need for the secondary cooling system in wintertime.

However, if the water heater is hot, if the heat pump is not running

to heat or cool the home and if the homeowner hasn't left instructions for the computer to turn on the appliances, then the computer decides to sell the energy back to the utility and routes the current from the roof through cables inside the house to a nearby substation, where the power will be rerouted to another, conventional home.

At night, when utility rates are about a third of their daytime cost, the home uses conventional electricity in the hot-water heater and storage-tank system, if this was not done during the day. The house also would be heated or cooled using conventional electricity but with a much lower utility cost.

"Spring and fall might be times of regular flow-back of power," says Cooper, who has written many such programs for industrial plants. "And if the family went on a month-long vacation, the utility bill for that month would probably show a substantial account credit."

All of these load-management functions are stored in the computer's internal memory, not on tapes or disks as many programs are. Thus, the decision-making continues 24 hours a day without the homeowner being aware of it. But this is not all Cooper wants it to do.

The computer also is interfaced (wired) with motion detectors and window alarms to provide a security system which can set off alarms both inside the house and at the local police station. Smoke detectors also are interfaced for this function. This is done through the computer's own dedicated telephone line which costs the homeowner the same as the monthly rate on any other telephone line.

While all this is going on, the homeowner may use the Commodore just as any other home computer. Through the phone line, he is connected to The Source and CompuServe, which allow him to get stock-market reports, make airline reservations, follow the national and international news and receive all types of information services. Network services offered by The Source include sending and receiving electronic mail, and the owner can even store personal information in a private Source file that's closed to all but the person with the password.

If he adds a printer, the homeowner also may use the computer for writing letters or doing paperwork at home instead of staying late at the office. And he can use the computer for home record-keeping and tax computation.

There are many ways in which the computer could assist the homeowner. As programmer Cooper says, "The technology is there, it's just a matter of getting it all working." For instance, the computer could be linked with a door lock to assist in burglar protection. A key in the lock would signal the computer to automatically disconnect the house burglar alarm, or the owner could enter a digital code into a wall box to do the same. The computer also could be programmed and linked to humidity sensors in the house, which would be quite valuable in the Southeast, where summer humidity can be as distressing as the sun's heat. With this use, the computer would keep track of the home's humidity level, and when it reached a certain upper limit, automatically turn on a dehumidifier.

The computer also could be hooked to sensors on the outside of the photovoltaic cells, which need periodic washing. It would take readings of dust accumulation and signal the owner when it is time to clean the solar panels.

The imagination takes it from there. Endless uses could be made of the computer; automated temperature and humidity control in a newborn's bedroom, automated garage and home-light-fixture turn-ons and turn-offs, automatic "lights-out" in the children's bedrooms—the list is endless, limited only by the homeowner's needs and patience with programming the computer.

The designers of Future I, however, can't insure that someone familiar with computers will own the house, so some of its features are duplicated. If the homeowner wants to set his thermostats to automatically turn down the heating or cooling at certain times of the day or when the outdoor temperature is at a certain level, he can do so by using either the computer or programmable thermostats with key panels and digital displays.

The computer may also be programmed to refuse to allow certain high-energy-consumption appliances to operate if there is no excess energy from the roof panels and it is during a high-utility-rate period.

But, says the program manager, anyone well-off enough to afford this home (Future I is expected to sell in the mid-$200,000 range) probably will not be willing to put up with too much inconvenience for the sake of saving a dollar. So if the program is operating, it may be overridden by a few keystrokes on the computer or by flipping a switch in the control room.

"People are used to walking up to boxes and switches to control things in a house. If the owner wants to do these things using the computer, he can. If he's more comfortable with flipping switches, he can do that, too," Cooper says.

Many of the house's features are passive solar, such as a Trombe wall for heat-retaining mass, phase-change salts and south-facing windows. On warm days, the owner will

not want these heating features. And without the computer, use of a feature like the roll-down shade under the overhang on the south side would depend on the owner's remembering to flip the switch—and being home to do it.

The computer also will be outfitted to automatically roll down those shades when the inside temperature reaches a certain limit.

While Future I is the first large, privately financed experiment with photovoltaics, using a regular family's lifestyle, it is not the first experiment with home computerization. In fact, many of its computerized functions already have been proven and are used regularly in industry.

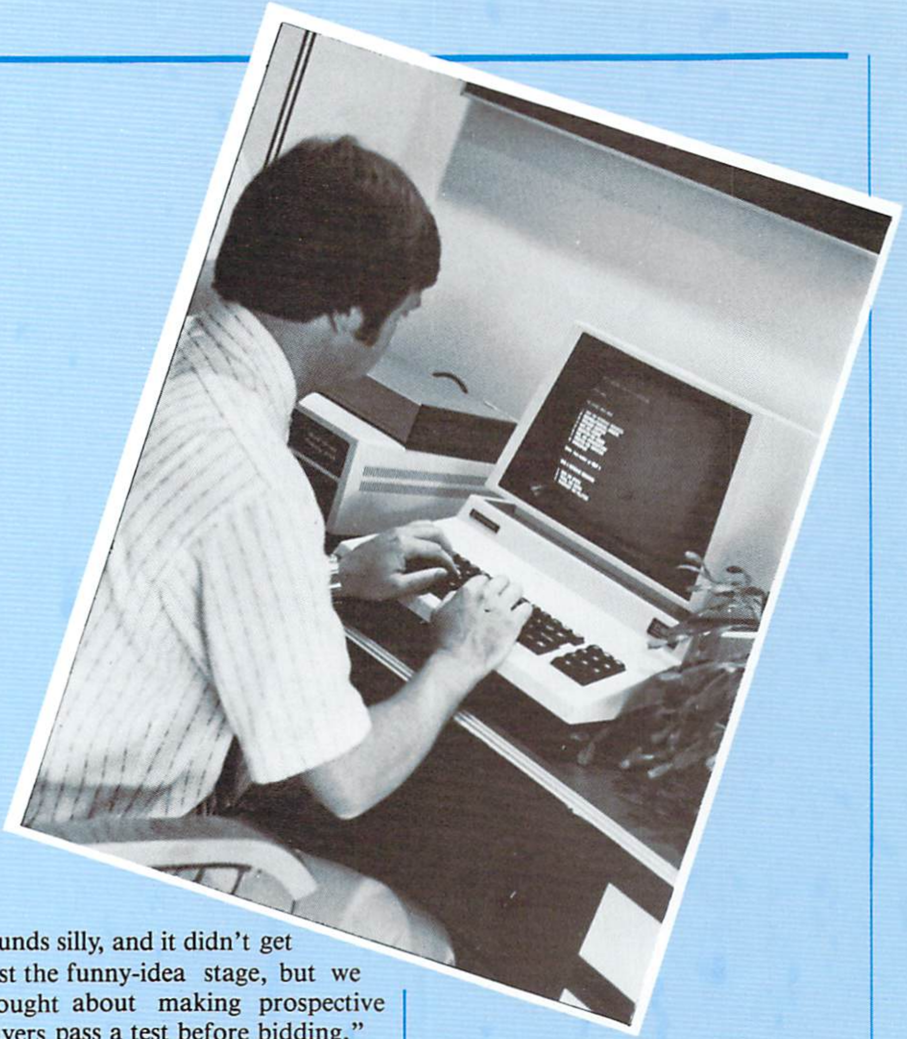
A few large corporations have experimented with home systems, too. Motorola's Semiconductor Group built a house from the ground up with computer control in mind in Ahwatukee, Arizona. It involved not only special appliances but several microcomputers to process all of its functions.

Honeywell's Residential Controls Group took a different approach, outfitting an existing home in Minneapolis with computers to determine what benefits could be gained from sophisticated controls alone, without the help of special appliances and architectural features.

Future I is different from both of these approaches in that its main computerized functions are to perform load management of the photovoltaic output.

And, says Cooper, "Those were very expensive experiments. Ours involves only one home computer (with two microprocessors) and off-the-shelf appliances."

Officials hope that someone both knowledgeable about passive solar installations and interested in microcomputers will buy the house. "This sounds silly, and it didn't get past the funny-idea stage, but we thought about making prospective buyers pass a test before bidding," Birdwell said. "Seriously, we would like to see someone get it who knows something about the things in the house. This is like a baby. If it cries, we would like someone in it who knows how to burp it."



The reason the human element is of concern to those connected with the design and possibilities for Future I is the fear that the prospective owner might not utilize the home's many features. If the new owner were intimidated by computers, he or she might not feel comfortable giving the computer commands which took full advantage of its energy-saving devices. For instance, the home could well become a hothouse in the summertime, if the owner were away from home all day and had not left instructions with the computer to roll down the shades to deflect sunlight from its expanse of glass, heat-saving wall and phase-change salts. Actually, the house would not overheat, but it possibly might use up all of its photovoltaic output and backup-cooling-system power and begin buying expensive daytime-rate electricity.

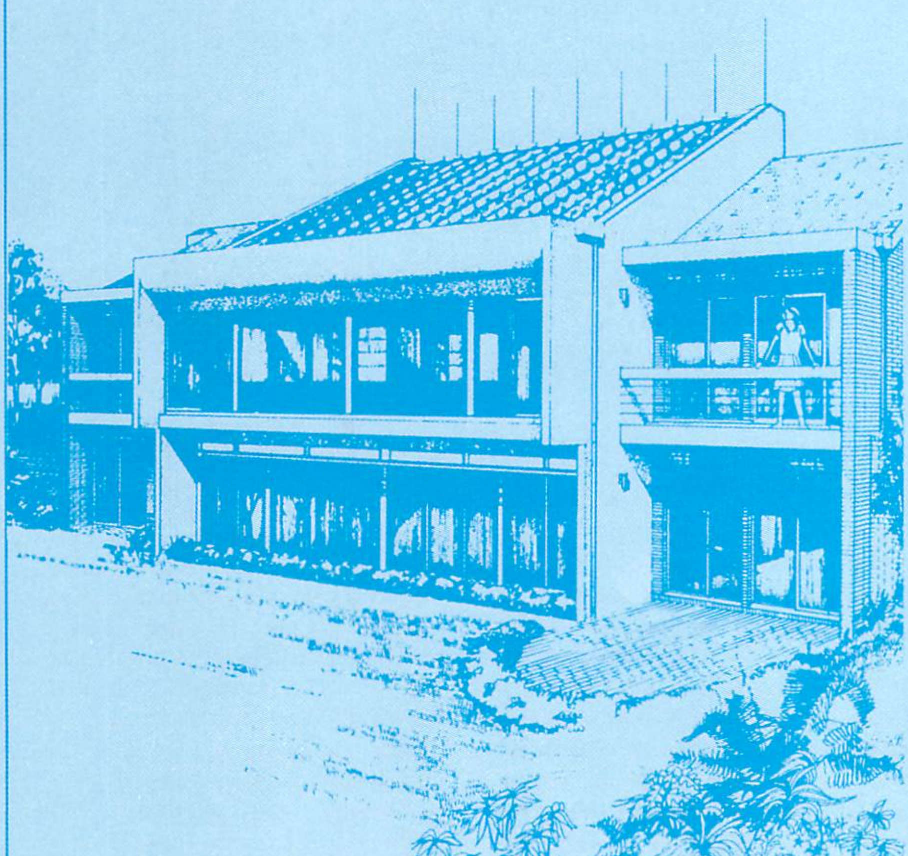
To avoid problems such as these, Cooper hopes to visit the home periodically to work with the new homeowners in determining the special computer uses most practical for their way of life.

The house incorporates features that in a decade may be commonplace. The passive solar features have been proven in use in the more than 60 houses in the Southeast that architect Sibly has designed. The cost of photovoltaics, although now outside the reach of most consumers, is expected to drop dramatically if increased-use projections are accurate. And the computerization?

"It won't be long until people come to view the computer as just another appliance in the home," Birdwell says.

PEGGY O'NEAL *teaches college courses in journalism and technical writing and writes extensively about science and technology.*

What Will Be Learned



The primary goal of Future I is to learn more about passive-solar features and how they contribute to the energy efficiency of homes. In the extreme North, these answers may be self-evident, for with its long, cold winters any *heat-producing features are welcomed. Similarly, in the Southwest the answers are easier to come by for there is nearly year-round sunshine. But what about more temperate climates, where swings from hot to cool are commonplace? And what of those questions, often stated as suppositions, that have never been tested?*

For instance, every architect and builder has an opinion as to whether the slab foundation of a home should be insulated. Some say the earth has a stable temperature at 10 feet underground. But the architect of Future I has done borings in the Atlanta area and has found that at 10 feet the earth's temperature varies by about 10° and does not stabilize until about 20 feet down.

What is the effect of this temperature variance on the energy usage of a home? To answer this and other questions, the designers of Future I have buried sensors 12 feet below the slab foundation of the house. Every 2 feet from there upward, there is another sensor. There are two sets of these; one under a portion of the slab which is not insulated and another under a portion which has 1 inch of polystyrene, 10 mils of polyethylene and 3 inches of sand as insulation. Readings from these sensors and from the floors on the lower level of the house should indicate whether foundation-slab insulation contributes to energy efficiency.

Sensors beneath the quarry-tile flooring in the family room also reveal much about that material's energy absorption. The flooring was installed under the supposition that, in the winter, it would absorb heat from the sun shining into the south-facing windows of the room and release it gradually throughout the day and night. During the summer, it would be shaded from the sun by a deck outside the second story. In theory, then, the quarry tile should remain cool during the summer.

Such temperature sensors have been installed in 69 places throughout the house and its grounds. There also are sensors in the ceiling, in the attic and in the downstairs controlroom area where they abound around the heat-pump heater/air conditioner.

The 69 sensors will take readings every 15 minutes, and these will travel through the house wiring to the control room. There, the information will be translated into pulses which automatically go by a separate telephone line to a microcomputer at the Georgia Institute of Technology where technicians will record it for further analysis. From this information, recommendations to home builders will be made.

Other questions to be answered by the sensors are:

- *What is the effect of extensive attic ventilation on cooling and heating?*
- *Will the phase-change salts in storage rods just inside the south-facing windows produce unwanted heat, even with the shades drawn?*
- *How much radiation will enter from the south-facing windows?*
- *Since the home's exterior is finished partly in brick and partly in stucco, which material will maintain a more constant indoor temperature?—P.O.*

The VIC Magician

Writing Games in BASIC

Part 2 . . . Moving Objects On the Screen

By Michael S. Tomczyk
Product Marketing Manager

In this issue, we continue our introduction to gamewriting in BASIC. Gamewriting is fun, but as we said in Part 1, there's no real "school" for gamewriters. So we're going teach you some of the skills involved . . . and let your imagination carry you forward from there. Ready? Let's go . . .

A Quick Refresher

Here's a quick refresher of some of the principles we learned last time (If you have trouble following this, the method for POKEing a character onto the screen is also described on Page 144 of your VIC owner's manual).

You're probably used to PRINTing VIC graphic symbols on your TV screen . . . but to create fast, effective animation, you should POKE the symbols onto the screen. POKEing symbols on the screen requires that you know the following things:

1. The SCREEN LOCATION (there are 506 locations, numbered from 7680 in the top left corner of the screen, to 8186 in the bottom right corner).
2. The POKE VALUE of the SYMBOL you want to display. The chart on page 141 of your VIC owner's guide shows these values. For example, the value for a solid ball is 81, a heart is 83, etc.
3. The COLOR LOCATION. This matches the screen location, but it's easy to calculate this number because the color location is ALWAYS the SCREEN LOCATION NUMBER ADDED TO THE NUMBER 30720.
4. The POKE VALUE of the COLOR you want to use. Color numbers are shown here:
   0-black 1-white 2-red 3-cyan 4-purple 5-green 6-blue 7-yellow

To help you plot screen locations for the animations we'll be working with in this series, we've provided a handy SCREEN LOCATION CHART with all the screen location numbers filled in for you. Imagine that this is your television screen, which has been divided up into 506 squares. Each square can contain any symbol on the VIC keyboard. To put a symbol in any one of the squares, all you have to do is tell the VIC which square (SCREEN LOCATION) you want to use and which symbol you want to display (POKE VALUE). Then you calculate the COLOR LOCATION (screen location plus 30720) and tell the VIC which color (POKE VALUE) you want to use.

To display any symbol on your screen, simply POKE the SCREEN LOCATION NUMBER, followed by a COMMA, and the POKE VALUE OF THE SYMBOL. Then POKE the COLOR LOCATION NUMBER (screen location plus 30720), followed by a comma, and the POKE VALUE OF THE COLOR. In a program it might look like this (enter this line):

```
10 POKE7680,83:POKE(7680+30720),4
```

Type the word RUN and press the RETURN key. You have just displayed a purple heart in the upper lefthand corner of the screen. Note that the number 7680 is the SCREEN LOCATION NUMBER for the space in the upper left space on your screen. The number 83 is the POKE VALUE of the heart symbol. The COLOR LOCATION is represented by adding 7680 plus 30720 in parentheses (You could have also added them together and used the number 38400). And the number 4 is simply the POKE VALUE of the COLOR purple.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
7680	7681	7682	7683	7684	7685	7686	7687	7688	7689	7690	7691	7692	7693	7694	7695	7696	7697	7698	7699	7700	7701
7702	7703	7704	7705	7706	7707	7708	7709	7710	7711	7712	7713	7714	7715	7716	7717	7718	7719	7720	7721	7722	7723
7724	7725	7726	7727	7728	7729	7730	7731	7732	7733	7734	7735	7736	7737	7738	7739	7740	7741	7742	7743	7744	7745
7746	7747	7748	7749	7750	7751	7752	7753	7754	7755	7756	7757	7758	7759	7760	7761	7762	7763	7764	7765	7766	7767
7768	7769	7770	7771	7772	7773	7774	7775	7776	7777	7778	7779	7780	7781	7782	7783	7784	7785	7786	7787	7788	7789
7790	7791	7792	7793	7794	7795	7796	7797	7798	7799	7800	7801	7802	7803	7804	7805	7806	7807	7808	7809	7810	7811
7812	7813	7814	7815	7816	7817	7818	7819	7820	7821	7822	7823	7824	7825	7826	7827	7828	7829	7830	7831	7832	7833
7834	7835	7836	7837	7838	7839	7840	7841	7842	7843	7844	7845	7846	7847	7848	7849	7850	7851	7852	7853	7854	7855
7856	7857	7858	7859	7860	7861	7862	7863	7864	7865	7866	7867	7868	7869	7870	7871	7872	7873	7874	7875	7876	7877
7878	7879	7880	7881	7882	7883	7884	7885	7886	7887	7888	7889	7890	7891	7892	7893	7894	7895	7896	7897	7898	7899
7900	7901	7902	7903	7904	7905	7906	7907	7908	7909	7910	7911	7912	7913	7914	7915	7916	7917	7918	7919	7920	7921
7922	7923	7924	7925	7926	7927	7928	7929	7930	7931	7932	7933	7934	7935	7936	7937	7938	7939	7940	7941	7942	7943
7944	7945	7946	7947	7948	7949	7950	7951	7952	7953	7954	7955	7956	7957	7958	7959	7960	7961	7962	7963	7964	7965
7966	7967	7968	7969	7970	7971	7972	7973	7974	7975	7976	7977	7978	7979	7980	7981	7982	7983	7984	7985	7986	7987
7988	7989	7990	7991	7992	7993	7994	7995	7996	7997	7998	7999	8000	8001	8002	8003	8004	8005	8006	8007	8008	8009
8010	8011	8012	8013	8014	8015	8016	8017	8018	8019	8020	8021	8022	8023	8024	8025	8026	8027	8028	8029	8030	8031
8032	8033	8034	8035	8036	8037	8038	8039	8040	8041	8042	8043	8044	8045	8046	8047	8048	8049	8050	8051	8052	8053
8054	8055	8056	8057	8058	8059	8060	8061	8062	8063	8064	8065	8066	8067	8068	8069	8070	8071	8072	8073	8074	8075
8076	8077	8078	8079	8080	8081	8082	8083	8084	8085	8086	8087	8088	8089	8090	8091	8092	8093	8094	8095	8096	8097
8098	8099	8100	8101	8102	8103	8104	8105	8106	8107	8108	8109	8110	8111	8112	8113	8114	8115	8116	8117	8118	8119
8120	8121	8122	8123	8124	8125	8126	8127	8128	8129	8130	8131	8132	8133	8134	8135	8136	8137	8138	8139	8140	8141
8142	8143	8144	8145	8146	8147	8148	8149	8150	8151	8152	8153	8154	8155	8156	8157	8158	8159	8160	8161	8162	8163
8164	8165	8166	8167	8168	8169	8170	8171	8172	8173	8174	8175	8176	8177	8178	8179	8180	8181	8182	8183	8184	8185

PAGE 1: SCREEN CHARACTER CODES

PROGRAMMER'S TIPS

PROGRAM NOTE—RANDOM NUMBER FORMULA: To make the VIC generate a random number which you can use in a program, you need to know the STARTING NUMBER and the RANGE of numbers you want the VIC to choose from. Here's what the random number formula looks like if you want the VIC to choose a number from a RANGE of 16 numbers from 10 to 25 (including the 10 and 25 in the range):

```
10 R=INT(16*RND(1))+10
20 PRINTR:FORT=1TO100:NEXT:GOTO10
```

The number 10 is the STARTING NUMBER. The number 16 is the RANGE of numbers inclusive from 10 to 25 (if you INCLUDE the 10 and the 25 in your range, then you actually have 16 numbers). The number 1 should always be enclosed as shown in parentheses. The letter R can be any variable. Line 10 actually chooses the random number. Line 20 just shows you how this number could be used. For example, we PRINTed the number (R), then inserted a TIME DELAY LOOP to keep the number on the screen, and after that we told the VIC to go back to Line 10 to get another number. The result is a series of random numbers from 10 to 25 appearing on your screen in no special order.

The Heartmaking Program

We'll start with a variation of a program we introduced last time. This program displays a heart, at random, in the top row on your screen (type the word NEW and press RETURN to erase any previous programs before typing this in):

(BEGINNERS: be sure to press the RETURN key at the end of each line to enter it!)

```
10 PRINT" SHIFT CLR/HOME ";
20 L=INT(23*RND(1))+7680:C=L+30720
30 POKEL,83:POKEC,5
40 GOTO20
```

Type the word RUN and press RETURN.

Briefly, this program begins by CLEARing the screen. Then it defines the variable L as a random number from 7680 to 7702 (the screen locations for the first row on the top of your screen). We then define the variable C as the COLOR LOCATION (C equals the screen location plus 30720, remember?). In Line 30 we POKE a heart (POKE VALUE 83) into the screen location, and POKE a color (green) into the color location. Finally, in Line 40 we go back to Line 20 and go through the process again. Notice that when you send the program back to Line 20, you are actually telling the VIC to get ANOTHER RANDOM NUMBER for the next screen location.

Erasing Symbols to Create Animation

The important point we want to introduce now is ERASING THE PREVIOUS CHARACTER. The illusion of animation is created by POKEing a symbol onto the screen, then POKEing another symbol next to it in the next square, and erasing the previous symbol. The way to erase a symbol is to POKE A SPACE into the symbol's screen location. The POKE VALUE for a space is 32. So if you want to erase a symbol which is being displayed in location L, just POKEL,32.

Type this line: **35 POKEL,32** (press RETURN)

Now RUN your program. Whoa! Too fast! You're displaying the heart at random on the screen, but it's being displayed and erased so fast that it's practically invisible. That's how fast computers work! So let's slow it down by inserting a TIME DELAY LOOP. Retype line 35 as shown:

35 FORT=1TO100:NEXT:POKEL,32 (press RETURN)

Now RUN your program. The time delay loop told the VIC to "count to 100" (computers count fast). So now, the VIC displays the heart once in the random location it chose in Line 20 . . . then it counts to 100 when it reaches Line 35 . . . so each heart stays on the screen for a count of 100 . . . and finally it erases the heart before going back to Line 20 to choose another location for the next heart.

Moving Symbols on the Screen

Now that you've had a little fun with hearts and learned how to erase symbols, let's take a look at how you can move a symbol around the screen. Here are some simple programs which show you how to do just that:

We're going to avoid random numbers in the next few sections and just show you how to move a symbol on the screen . . . presuming that you know the starting point and the destination.

Horizontal Movement

Pick a location on the left side of the screen. Let's choose location 7966 and move a purple ball from the left side to the right side of the screen. The secret is that we ADD ONE to the screen location to make any object move from LEFT TO RIGHT. Type the word NEW and press

RETURN, then enter this program:

```
10 PRINT" SHIFT CLR/HOME ";:L=7966:
   C=L+30720
20 POKEL,81:POKEC,4
30 L=L+1:C=C+1
40 FORT=1TO100:NEXT
50 GOTO20
```

Type the word RUN and press RETURN. Wait . . . this program isn't really animating the ball. It's just displaying the ball over and over again from left to right. How do we make ONE ball move? We erase the previous ball! Good . . . now hold down the RUN/STOP key and press the RESTORE key at the same time. This interrupts the program without erasing it. Now type this line (it will be automatically added to your program):

35 POKEL-1,32 (press RETURN)

Now type the word RUN and press RETURN to see it work. Presto! You are now animating a ball . . . but . . . it keeps going and going and going. How do you make it stop? How do you make it move to a certain point and stop? To do this you need to insert an IF...THEN statement. Type this line:

45 IFL=7987THENEND (press RETURN)

Now RUN your program. The ball moves to the last space on that line and stops. That's because you told the VIC in line 35 that IF the location becomes 7987 (the screen location number of the last space on that line) THEN END the program. Type LIST and press RETURN to see the full program, then cursor up to the number 7987 and change it to 7980, then press RETURN. Now cursor down to a blank area on your screen, type the word RUN and press RETURN. Your ball only moves to screen location 7980 and stops. This is how you move a ball horizontally from one location to another.

When you LIST your program it should look like this. In a moment, we'll show you a shorter way to do the same thing we just did but for now, so you understand exactly what we're doing, let's dissect this program line by line and see what's happening here:

```
10 PRINT" SHIFT CLR/HOME ";:L=7966:C=L+
   30720
20 POKEL,81:POKEC,4
30 L=L+1:C=C+1
35 IFL=7980THENEND (press RETURN)
40 FORT=1TO100:NEXT
45 POKEL-1,32 (press RETURN)
50 GOTO20
```

Line 10 CLEARs the screen. Then we define the variable L as screen position 7966 and variable C as the matching color location. From now on we can use L for screen position and C for the matching color location.

Line 20 is where we POKE the ball symbol and its color onto the screen. The first time the VIC goes through this line, it will put the ball at location 7966 because at this point in the program, L equals 7966.

Line 30 is where we move the ball forward, by adding a value of ONE to both the screen and color locations . . . so L now equals L plus one and C now equals C plus one.

Line 35 is our IF...THEN statement which tells the VIC to END the program if the value of L equals 7980.

Line 40 is the TIME DELAY LOOP that keeps the ball displayed on the screen for a "count" of 100. You can change the number 100 to make the ball go faster or slower.

Line 45 ERASES the previous ball by POKEing a BLANK SPACE (value 32) into the location immediately BEHIND (L-1) the location you're now using.

Line 50 tells the VIC to go back to Line 20. Now we start over again and this time the VIC POKEs the ball into the next space to the right of the last position we used (because L now equals L plus one). The program will keep cycling through and going back to Line 20 over and over again until L equals 7980 and the program ENDs.

A Shorter Move Program

Here's a shorter way of doing what we just did:

```
10 PRINT" SHIFT CLR/HOME ";
20 FORL=7966TO7987:C=L+30720:POKEL,81:
   POKEC,4:FORT=1TO100:NEXT
30 POKEL,32:NEXT
```

Type RUN and press RETURN. The ball moves all the way across the screen and disappears. Here's a variation on the same theme, which keeps the ball on the screen when it moves to the last space:

```
10 PRINT" SHIFT CLR/HOME ";
20 FORL=7966TO7987:C=L+30720:POKEL,81:
   POKEC,4:FORT=1TO100:NEXT:POKEL,32
30 POKEL-1,81:NEXT
```

Moving From Right to Left

The movement routine for moving from right to left is the same as left to right, except instead of ADDING

ONE to the screen and color locations, you SUBSTRACT ONE (the STEP-1 does this). Also, the screen locations start on the right and move left. Here's an example:

```
10 PRINT"SHIFT CLR/HOME";
20 FORL=7987TO7966STEP-1:C=L+30720:
   POKEL,81:POKEC,4:FORT=1TO100:NEXT
30 POKEL,32:NEXT
```

Moving From Top to Bottom

The movement routine for moving from top to bottom is the same, except the secret here is that you must ADD TWENTY-TWO to the screen and color locations. Here is a program which starts the ball at location 7690 and moves vertically to the bottom of the screen:

```
10 PRINT" SHIFT CLR/HOME ";
20 FORL=7690TO8174STEP22:C=L+30720:
   POKEL,81:POKEC,4:FORT=1TO100:NEXT
30 POKEL,32:NEXT
```

Moving Bottom to Top

This program does just the opposite of the previous program. It starts at the bottom and moves UP by SUBTRACTING TWENTY-TWO from the screen location:

```
10 PRINT" SHIFT CLR/HOME ";
20 FORL=8174TO7690STEP-22:C=L+30720:
   POKEL,81:POKEC,4:FORT=1TO100:NEXT
30 POKEL,32:NEXT
```

Moving Diagonally

It stands to reason that if adding 22 makes a symbol move down, then ADDING 23 makes the symbol move DOWN DIAGONALLY TO THE RIGHT, ADDING 21 moves it DOWN DIAGONALLY TO THE LEFT. SUBTRACTING 23 moves it DIAGONALLY UP AND TO THE LEFT and SUBTRACTING 21 moves it DIAGONALLY UP AND TO THE RIGHT. Here's a program to move the ball DIAGONALLY UP TO THE LEFT:

```
10 PRINT" SHIFT CLR/HOME ";
20 FORL=8174TO7690STEP-23:C=L+30720:
   POKEL,81:POKEC,4:FORT=1TO100:NEXT
30 POKEL,32:NEXT
```

Notice in this program that the ball moves diagonally from the bottom center of the screen, to the left edge of the screen . . . but then it reappears at the far right edge and continues up diagonally off the top of the screen! To eliminate this, you have to set a LIMIT for the moving symbol. An IF . . . THEN statement takes care of this. Simply add Line 25 by typing it, then RUN your new program:

```
25 IFL<7965THENEND (press RETURN)
```

Line 25 tells the VIC to check to see if L is 7965. When the screen location hits location 7965, the program ends.

Combining Movements

You can combine some of the movements we've shown you, by using IF . . . THEN statements to set up conditions that will change the symbol's direction. First, let's start with a program that makes the ball "bounce."

```
10 PRINT" SHIFT CLR/HOME ";
20 FORL1=8174TO7690STEP-22:C1=L1+30720:
   POKEL1,81:POKEC1,4:FORT=1TO100:NEXT
30 POKEL1,32:NEXT
40 FORL2=7690TO8186STEP22:C2=L+30720:
   POKEL2,81:POKEC2,4:FORT=1TO100:NEXT
50 POKEL2,32:NEXT
```

If you like, you can insert sound effects. Here's a possibility . . . insert these two "sound effect" lines in the program above:

```
25 POKE36878,15:FORS=150TO250STEP10:
   POKE36875,S:NEXT:POKE36875,0
55 POKE36875,150:FORD=1TO100:NEXT:
   POKE36875,0
```

See how easy it is to turn a simple program into an ILLUSION? That's what we mean when we talk about "VIC MAGIC." Ask the VIC magician! You've just become a magician yourself . . . you made a ball climb to the top of the screen, fall back down (it falls slowly because the screen atmosphere is heavy) and go "blonk."

Try some different sound effects. You can also make the ball move faster by playing with the number 100 in the timing loop in Lines 20 and 30. For example, change the number 100 in Line 30 to the number 50 and the ball will fall twice as fast as it rose.

Now let's make one object react to another:

```
10 PRINT" SHIFT CLR/HOME ";
20 FORL1=7690TO8174STEP22:C1=L1+30720:
   POKEL1,81:POKEC1,4:FORT=1TO100:NEXT
30 FORL2=8174TO7690STEP-22:C2=L2+30720:
   POKEL2,81:POKEC2,4
40 FORT=1TO100:NEXT:POKEL2,32
50 NEXTL2:NEXTL1:POKEL1,32:POKEL2,32
```

Does this movement suggest a game? We're getting closer. Next time we'll talk about how we can combine some of these movements and use the keyboard and joystick controls to make a real game. In the meantime, keep experimenting. If you come up with anything interesting that you'd like to share with the VIC MAGICIAN's readers, send it in and we'll print the best entries in this column.

READ . . . DATA

By Michael S. Tomczyk

You've probably used the READ and DATA statements in your music programs. These BASIC instructions are mostly used when working with large amounts of information, or "data." The format consists of putting data information in a line, usually at the END of your program, with each piece of information separated by a comma. One or more READ statements are then included in the main part of your program. Here's an example that READs five numbers from a DATA line and PRINTs them:

```
10 READ A
20 PRINTA;
30 GOTO10
40 DATA1,2,3,4,5
```

LINE 10 READs the data from the DATA statement in LINE 40. In this case it READs the number 1 (note the variable A can be ANY variable you choose).

LINE 20 PRINTs the number A which was read in Line 10.

LINE 30 goes back to Line 10, but the SECOND time the program goes back and READs data, it reads the SECOND number (2). The THIRD time it READs data it reads the THIRD number (3), and so on.

LINE 40 contains the data which is read in Line 10. Each piece of data is separated from the next data by a comma. Note that the data is ALWAYS read in the order you put it in the DATA statement.

When you RUN this program it will execute but when finished it will give you an ERROR4 IN 10 message, which means that the READ statement was reading the data and it ran out of data. You can solve this with a new Line 35 and 40:

```
35 IF A = -1 THEN END
40 DATA1,2,3,4,5,-1
```

This program uses both numeric and string variables in an unusual way. Notice that words, phrases and other string information don't have to be enclosed in quotation marks when listed in a DATA statement.

```
10 READX$:READA:READB
20 IFA = -1THENEND
30 PRINTX$A"IS $"B
40 GOTO10
50 DATACAR EXPENSE,1,40, CAR EXPENSE,2
60 DATA120,ELECTRICITY,1,100,
   FOOD EXPENSE,1,50
70 DATA -1, -1, -1
```

LINE 10 READs 3 separate pieces of information, IN ORDER, from the DATA statements in Lines 50–70.

LINE 20 sets up the end of the program so you don't get an error message when you run out of data.

LINE 30 uses the information which you READ in Line 10. Here we're using the information in a PRINT statement which begins with the string information (in this case one or two words) represented by X$ and then uses the numbers represented by A and B.

LINE 40 goes back to Line 10 to READ the next 3 pieces of DATA. The DATA is always read in order. Your DATA can spill over from one line to the next like it did in Lines 50–60 and the program will continue reading the DATA in order, but the first word in every DATA line must always be DATA.

LINES 50–70 contain the DATA being read and used in the program. The DATA is always read in order . . . here the information is organized in groups of 3. Each group of 3 items is read in order, used in the program, and then the next group of 3 is read. You can reset everything and make it go back to the beginning by using the CLR command (see BASIC INSTRUCTION GLOSSARY). The last line lets the program END without an error message.

Summary

This quick explanation of READ . . . DATA is designed to give you enough information so you can do some experimenting on your own. Using DATA in your programs is one of the keys to writing practical programs that let you store information, a subject we'll cover more fully in a future VIC MAGICIAN.

Speed Up Your BASIC Programs

by Andy Finkel

Video games generally need fast response and smooth graphics. That's why very few video games are written in BASIC. . . . machine language is much faster (if not as easy). But almost everyone *knows* BASIC, and most of those people want to write games. This article was written to help those people wring the last bits of speed from any Commodore computer. (The techniques work quite well in non-game programs also.)

I can hear you now: So, I can make my BASIC program run much faster . . . but what's the catch? Well, the catch is that your programs will be hard to read. Many of the methods produce obscure-looking code. Six months from the time you write it, you will have a hard time figuring out the logic of the program. If you are willing to put up with this, you can have faster programs.

1. Place all subroutines at the beginning of your program, and put in a line before the subroutines to jump to the actual start of the program when RUN is typed.

When BASIC sees a GOSUB line number, it starts looking for the line number at THE BEGINNING OF THE PROGRAM. If subroutines are first, they will be found faster.

```
EXAMPLE: 1 GOTO 100
         10 REM SUBROUTINE START
```

2. No spaces anywhere.

Spaces are just characters that BASIC is going to have to ignore. Ignoring spaces takes extra time.

```
EXAMPLE: 1 PRINT 7 + 9 / (3↑7.9)
         2 PRINT 7+9/(3↑7.9):REM FASTER
```

3. No REM statements.

The same objection holds for REM statements as for spaces.

4. Use single letter variables.

It takes longer to read the extra character, as well as using more memory in program lines.

5. Cram as much as possible onto each line by using colons to create multiple statements.

This reduces the number of line numbers, each of which takes up 5 bytes, speeding up GOTOs. When a branch instruction like GOTO and GOSUB is executed, the computer starts at the first line and hunts line by line until it finds the right number.

6. Number your program by 1's.

This will reduce the number of digits that BASIC will have to read on GOTO statements.

7. Never use integer variables.

When BASIC encounters an integer variable, it converts it into floating point. When faced with a statement like A% = B% + C% BASIC must convert B% and C% into floating point, perform the addition, then convert the floating point value into an integer to put into A%. All this takes time, not to mention the extra character (% sign) that BASIC must read.

8. Set variables to the constants you need in your program.

BASIC must convert a constant into floating point format to use it. If the constants are already converted (in variables) there will be a speed improvement.

```
EXAMPLE: 10 POKE 36879, 8: REM NO!
         20 POKE Q, J: REM FASTER
```

9. Do not let BASIC do a garbage collection whenever it wants.

Garbage is memory space used up by string and calculations operations. Programs build up garbage, which must eventually be cleared out.

The moment BASIC chooses for its garbage collection may not agree with the time you want a garbage collect to occur. Force garbage collection at non-critical times by executing a FRE(O) statement.

10. PRINT graphics to the screen rather than POKEing them.

For VICs and 64's, the PRINT statement puts the color on the screen as well as the character, which takes 2 POKES.

11. Never say IF(   )THENGOTOlinenumber.

The GOTO is unneeded.

12. Set an initial value for ALL floating point variables at the beginning of the program, before any arrays are dimensioned or strings are defined.

When new variables are created, BASIC must move all the arrays to make room, which is time consuming.

13. Put frequently used variables first in the initial value statements.

BASIC will find the values of those variables first.

14. Never use a variable name in a NEXT statement.

When BASIC sees a variable name after a NEXT, it is forced to check for a match. If there is just a NEXT with no variable name, BASIC will assume it is correct and continue execution.

15. Use machine language routines.

There is a limit to how fast BASIC can go. Machine language has a much higher top speed. Short machine language routines can really help the performance of a BASIC program.

PROGRAMMER'S NOTEBOOK
Fun With Prime Numbers

by
Neil Harris

One day, when lost in New York, I asked a passerby, "What's the best way to get to Carnegie Hall?"

The answer: "Practice!"

This isn't just a joke, it is good practical advice for learning anything, be it music, writing, or programming. You can't learn programming by reading books, or running other people's software, or even by examining programs. These are all ways to achieve understanding, but imagine trying to learn to play music by reading books, listening to records, and examining scores. No amount of study replaces practicing with the instrument.

This column, therefore, won't get heavily into the theory behind specific BASIC commands, nor will it dissect a large program into individual steps. I will try to help you grow from a novice programmer to an expert by discussing techniques and different approaches to a problem.

The biggest hurdle is to go from understanding each BASIC command to putting them all together in a large program. If you know how to write a paragraph, you aren't a novelist.

Every programming problem has an infinite number of solutions. There is no best way, although there may be a fastest way, a shortest way, or a most user-friendly way. You must keep many goals in mind when programming, only one of which is the specific problem.

Take, for example, a simple mathematical program. One of the easiest types of numbers to play with are prime numbers. For those of us who forget our math, a number is prime if it is not divisible by a whole number, other than itself and 1. The first few prime numbers are 2, 3, 5, 7, 11, and so on. The number 4 isn't prime because it is evenly divisible by 2. The only even number that is prime is 2, because every other even number is divisible by 2.

How would a program decide if a number is prime? The most simple approach is to divide the number by every number between 2 and the number. Program 1 is a simple program that displays prime numbers larger than 2.

PROGRAM 1:

```
0 REM SLOW PRIMER, VERSION 1, BY --> NEIL HARRIS
10 N=3
20 FOR L=2 TO N-1
30 IF N/L = INT(N/L) THEN 60
40 NEXT L
50 PRINT N,
60 N=N+1
70 GOTO 20
```

The program does the job, with only one real problem: it is very slow. It takes 14 seconds to solve for all primes less than 100, and the time goes up geometrically as the number tested gets larger. So this program would be perfectly acceptable for small prime numbers, but deadly for big ones.

How can this program be improved? In many ways. First, we already know that no even numbers can be prime (except 2), but the program checks all numbers regardless. The first improvement is to change line 60 to N = N + 2, skipping all the even numbers.

This is a minor change, since the even numbers are rejected on the very first division every time. You'll find that the program doesn't really speed up much.

There is another rule of primes that comes in handy here. Once you've divided the number by anything less than the square root of the number, you're safe to assume it's prime. You are guaranteed to find any non-primes by this point.

We can change line 20 to FOR L = 2 TO SQR(N). This saves much time. Now the same program only takes about 6 seconds to check all the numbers up to 100.

There is one last touch to add for speed. In line 30, the program is performing the same division operation twice. If we first put the results of the division into a variable, then just use the variable, this program will be as fast as possible. Change this to 30 Q = N/L: IF Q = INT(Q) THEN 60. The new version of this program is Program 2, below. It takes about 4 seconds

to get to 100. This is 3.5 times faster, and the time savings is even more dramatic for higher numbers.

PROGRAM 2:

```
0 REM SLOW PRIMER, VERSION 2, BY --> NEIL HARRIS
10 N=3
20 FOR L=2 TO SQR(N)
30 Q=N/L: IF Q=INT(Q) THEN 60
40 NEXT L
50 PRINT N,
60 N=N+2:GOTO20
```

Keep in mind that both programs perform exactly the same task, finding prime numbers. The difference is strictly in program efficiency.

Is this now the best solution to the prime number problem? Not really. At this point we've narrowed down the number of calculations necessary to discover primes, but not down to the bare minimum. We've been dividing by every odd number up to the square root of the number being tested.

A faster test would be to divide only by prime numbers. Among the 50 odd numbers between 1 and 100, only 22 are prime. If we had a way to check only the primes, we would improve the program's efficiency by over 50%.

What's the best way for checking only primes? Certainly not by re-calculating the primes for each new number. Since the program discovers all primes from the beginning, it is easiest to just store each prime in an array. The number being tested can then be divided by every prime less than the number's square root, ignoring any numbers that aren't prime. Program 3 does this.

PROGRAM 3:

```
0 REM FAST PRIMER, VERSION 1, BY --> NEIL HARRIS
10 DIMA%(1000)
20 X=1:A%(0)=2:A%(1)=3:N=3
30 N=N+2:Q=SQR(N):FORL=0TOX:IFA%(L)>QTHENX=X+1:A%
   (X)=N:PRINTN,:GOTO30
40 J=N/A%(L):IFJ=INT(J)THEN30
50 NEXT
```

This program is a little tricky, so I'll spend some time explaining its mechanics before going on.

The prime numbers are held in the array A%, which is first "seeded" with the first 2 primes. The integer array is used because it only takes 40% of the memory that a regular, floating-point array would use. All primes are integers, anyway.

The variable X is a pointer. It holds the number of primes in the array. When a new prime number is discovered, the X value is incremented (one is added to its value), and the new prime is added to the list.

ARRAY A% at start

A%(0)	A%(1)	A%(2)	A%(3)	A%(4)	A%(5) . . .
2	3				
	↑ X = 1				

ARRAY A% after 2 new primes added

A%(0)	A%(1)	A%(2)	A%(3)	A%(4)	A%(5) . . .
2	3	5	7		
			↑ X = 3		

The FOR-NEXT loop never is completed. The loop is terminated prematurely by either one of 2 conditions: the number is found to be not prime, in which case the next number is selected; or the prime number to be divided is found to exceed the square root of the number being tested, in which case the tested number is added to the list of primes. The loop's old values are erased by starting a new loop using the same variable name (L).

This program maintains a higher speed than program 2, especially noticeable as the numbers get larger.

Are we finished with the prime number project yet? Would I ask a silly question like that if we were?

Programmers have a saying: programs are never finished. There is always some new feature, some improvement, some tinkering left to add. Let me fill you in on the background of these programs before continuing with the technical stuff.

I learned to program in BASIC in 1970. One of the first programs I wrote was the version shown as program 1. By the next year, I had perfected the program to the last one shown above. Then it lay dormant for another year. Then I heard a mathematical proof that inspired more sophisticated tinkering with the program.

Is there an infinite or finite number of prime numbers? If you run the prime number program for long enough, you'll see that the primes tend to become farther and farther apart as they grow larger. Might there be a point when there are no more prime numbers? ▶

Let's assume that there is a finite number of prime numbers. Multiply all the prime numbers together. Add one to the result. The final number is not divisible by any of the prime numbers, and is therefore prime itself. That is proof for there being an infinite number of primes.

This proof has an intriguing side effect. Starting with the first prime number, you can multiply all the primes together, add 1, and the result is prime. 2+1=3, 2*3+1=7, 2*3*5+1=31, etc. I don't know if the result must be prime, and I would certainly be interested in finding out. I just know that it works as far as I can test.

The world's record for the largest prime number discovered is something like 2 to the 32,000 power (my copy of the Guinness Book of World Records may be out of date by now). While a very large number, it isn't hard to multiply enough numbers together to exceed that value.

I then wrote a program, to multiply primes together and print the result. Unfortunately, most computers are only accurate to ten or so decimal places, and accuracy is quickly lost. I needed a new technique.

The method that worked was to store each digit of the result in an array. You can then multiply each digit by a number, then perform the necessary carry operation to create a good result with just one digit in each element of the array. This program was tested in 1975 on Cornell University's computer by Andy Finkel, who is also at Commodore now. He let the program run for a whole day, and the printout of the final number was the size of a phone book. We never did find out if it was really prime, but it ran to hundreds of thousands of digits. If this method really works to discover primes, then we broke the record by many orders of magnitude.

Program 4 is a Commodore BASIC version of this large-number generator.

PROGRAM 4:

```
0 REM PRIME MULTIPLIER,VERSION 1,BY -->NEIL HARRIS
10 DIMA%(1000),B%(5000)
20 X=1:A%(0)=2:A%(1)=3:N=3:B%(0)=6:D=0
30 N=N+2:Q=SQR(N):FORL=0TOX:IFA%(L)>QTHEN60
40 J=N/A%(L):IFJ=INT(J)THEN30
50 NEXT
60 X=X+1:A%(X)=N:Q=0:M=0:PRINTN,
65 B%(M)=B%(M)*N+Q:Q=INT(B%(M)/10):B%(M)=B%(M)-10*Q
70 M=M+1:IFM<=DTHEN65
80 IFQ>0THEND=D+1:GOTO65
90 FORM=DTO1STEP-1:PRINTCHR$(48+B%(M));:NEXTM:PRINT
   "1":GOTO30
```

The B% array holds the digits of the result, with the lowest digit in element 0. The variable D is the pointer for this array, telling how many digits are in the result. The lines up to number 50 are the same as the previous program, but once the prime number is discovered the program goes on to multiply it by the B% array. The variable Q holds the carry from the previous multiplication.

ARRAY B% at start

B%(0)	B%(1)	B%(2)	B%(3)	B%(4)	B%(5). . .
6					

↑
D=0

ARRAY B% after 2 primes

B%(0)	B%(1)	B%(2)	B%(3)	B%(4)	B%(5). . .
0	1	2			

↑
D=2

Line 90 prints the result. The last digit of the multiplication is always 0, so we can ignore it and just print a 1.

Does it stop here? You should know me better than that by now.

Program 5 adds one cosmetic feature: it puts commas in the number displayed, which makes it easier to read.

PROGRAM 5:

```
0 REM PRIMER, VERSION 3, BY --> NEIL HARRIS
10 DIMA%(1000),B%(5000)
20 X=1:A%(0)=2:A%(1)=3:N=3:B%(0)=6:D=0
30 N=N+2:Q=SQR(N):FORL=0TOX:IFA%(L)>QTHEN60
40 J=N/A%(L):IFJ=INT(J)THEN30
50 NEXT
60 X=X+1:A%(X)=N:Q=0:M=0:PRINTN,
65 B%(M)=B%(M)*N+Q:Q=INT(B%(M)/10):B%(M)=B%(M)-10*Q
70 M=M+1:IFM<=DTHEN65
80 IFQ>0THEND=D+1:GOTO65
90 FORM=DTO1STEP-1:PRINTCHR$(48+B%(M));:IFM/3=INT
   (M/3)THENPRINT",";
100 NEXTM:PRINT"1":GOTO30
```

This is a little easier to understand, but for my last version of the program I decided to go all the way for readability. Instead of printing out the digits of the result, this program translates the numbers into English. Where program 5 shows the number "31", program 6 shows the words "thirty one". It can handle numbers up to octillions, and could go even farther if I had any idea what the next order of magnitude is called. The only suffix beyond octillions that I know off-hand is a googol, which is 1 followed by 100 zeroes, but I don't know the ones in between.

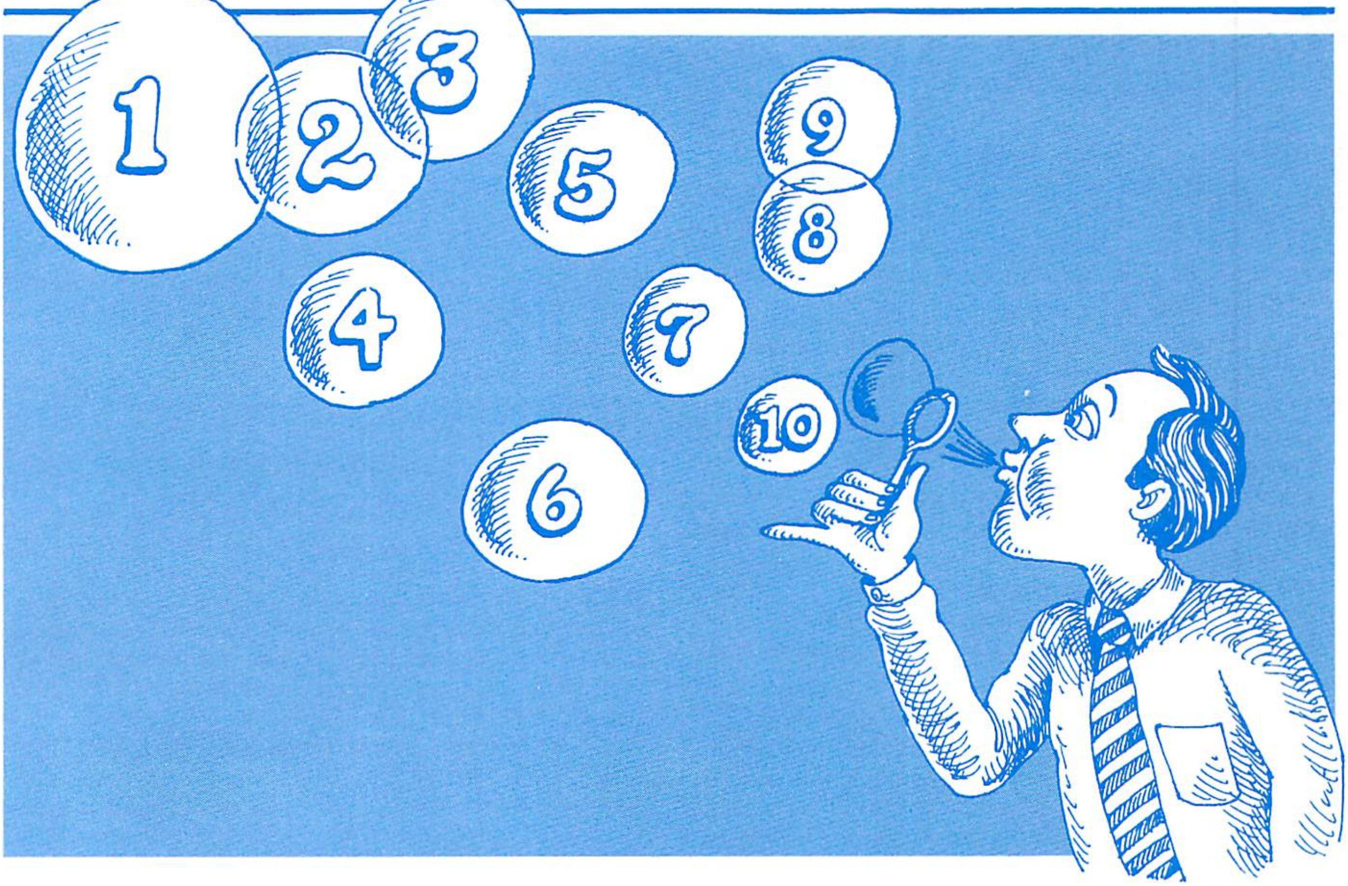
PROGRAM 6:

```
0 REM PRIME MULTIPLIER, VERSION 3, BY -->
  NEIL HARRIS
5 FORL=1TO9:READA$(L):NEXT:FORL=2TO9:READB$(L)
  :NEXT:FORL=0TO9:READC$(L)
6 IFL>2THENC$(L)=C$(L)+"TEEN"
7 NEXT:FORL=1TO8:READD$(L):IFL>1THEND$(L)=D$(L)
  +"illion"
8 NEXT
10 DIMA%(1000),B%(5000)
20 X=1:A%(0)=2:A%(1)=3:N=3:B%(0)=6:D=0
30 N=N+2:Q=SQR(N):FORL=0TOX:IFA%(L)>QTHEN60
40 J=N/A%(L):IFJ=INT(J)THEN30
50 NEXT
60 X=X+1:A%(X)=N:Q=0:M=0:PRINTN,
65 B%(M)=B%(M)*N+Q:Q=INT(B%(M)/10):B%(M)=B%(M)-10*Q
70 M=M+1:IFM<=DTHEN65
80 IFQ>0THEND=D+1:GOTO65
90 B%(0)=B%(0)+1:FORL=INT(D/3)TO0STEP-1:A=B%(L*3+2)
  :B=B%(L*3+1):C=B%(L*3)
100 IFA>0THENPRINTA$(A)" HUNDRED ";
110 IFB>1THENPRINTB$(B)"TY ";
120 IFB=1THENPRINTC$(C)" ";:GOTO140
130 IFC>0THENPRINTA$(C)" ";
140 IFA+B+C>0THENPRINTD$(L)" ";
150 NEXT:B%(0)=B%(0)-1:PRINT:GOTO30
200 DATAONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,EIGHT,NINE
210 DATATWEN,THIR,FOR,FIF,SIX,SEVEN,EIGH,NINE
220 DATATEN,ELEVEN,TWELVE,THIR,FOUR,FIF,SIX,SEVEN,
    EIGHT,NINE
230 DATATHOUSAND,M,B,TR,QUADR,PENT,SEXT,SEPT,OCT
```

By the way, while I was programming this last version, I ran into a snag. I was trying to use a tricky approach that involved taking the number one digit at a time, instead of three digits as the final version did. The program was full of bugs, and each time I solved one problem another one appeared. After a solid hour of work, the program was so convoluted that I was having trouble following the logic. Finally, I resorted to a time-tested programming technique: I erased the program and called it a day. The next day, I started again, trying a more straightforward approach. Within 15 minutes, I had it working. Sometimes it's best to scrap a bad approach and start fresh. I don't call that wasting time, I call it a learning experience. Once you've tried to solve a problem, even if you fail, you've learned enough about the problem to be able to succeed later.

Is this program really useful? Not really, as it relates to primes. But there are applications when you need to spell out the numbers in English. How about a program that prints checks?

The moral of all this is that any technique that you learn in programming is bound to apply to some problem at some time. Writing programs is, after all, the only way to learn.



S.O.S. Revisited
(Some Other Sorts)

by Dale DePriest

I really enjoyed the pair of articles written by Dwight Wheeler on boundary bubble sorts and shell sorts (August 1982 *Commodore* magazine). They were very well done and really served to whet my appetite in this whole sorted affair. One of the things that interested me was the fact that a reasonably good sort could actually be squeezed into four or five lines of code. (You can do it with these sorts if you use multiple statements per line.)

To understand the rest of this article you may find it helpful to review the material in Mr. Wheeler's article. Try out his sorts and get a feel for the background of this article. His article is really good reading but if you can't get a copy you can still use the sort that is presented in this one.

First I set out to see just how fast these sorts were. I looked around and found a reputably fast sort algorithm called a quick sort to compare them with. The quick sort is a pretty complicated sort technique that uses a second array to keep track of where you are in the first. Low and behold even tho' it was considerably more complex and longer routine the quick sort was still the fastest. It was almost twice as fast as the bbsort and over 20% faster than the shell sort when running Mr. Wheeler's benchmark. The quick sort has its own Achille's heel, however, in that it is much slower on lists that are already partially ordered. Aha! Perhaps if we can just figure a way to speed up the shell sort just a little perhaps we can get a universal sort.

Using the premise of simple must be better let's see if we can improve the shell sort while not giving up too much of its simplicity. What is needed is a method of improvement that is as elegant as the boundary solution is to the simple bubble sort. That's it! What we need is a boundary shell sort.

If you take a look at the shell sort you'll find that it really consists of two levels of sort. The outside level slices the pie into smaller and smaller pieces until there aren't any left. The inside loop tries to sort the piece of pie to the best of its ability with a technique that resembles the bubble sort. This resemblance is the answer we needed so we'll apply the boundary technique to the inner loop.

The theory goes something like this. Since we go through the loop from the bottom the last swap made in a pass means that nothing larger than this needs swapping. So, why bother looking at it anymore and as a matter of fact you don't have to look at the last thing swapped any longer either. However be careful, once we enter the

PROGRAMMER'S TIPS

outer loop all bets are off and we'd better check everything again.

With our theory well in hand let's dive in and analyze this sort. Our demo program will use Mr. Wheeler's data array for testing, so let's read it in.

```
100 DIM A$(100): REM Dimension the array
110 N = 1: REM set the counter
120 READ A$(N): REM read the data
130 IF A$(N) = "END" THEN 160: REM check for
    end
140 N = N + 1: REM increment the counter
150 GOTO 120: REM loop until all items are read
160 N = N - 1: REM the last item doesn't count
```

Now we can begin the actual sort process. The data is Mr. Wheeler's benchmark data i.e. the alphabet in the order it's listed on a standard keyboard. You can't get much more mixed up than that. We begin by going to the end. This simple ruse saves checking for the same thing in two different places. This time we're checking to see if there is really a list long enough to be worth sorting.

```
190 REM . . . BOUNDARY SHELL SORT . . .
200 G = N: GOTO 300: REM don't loose track of N,
    we'll need it later.
```

We'll get to 300 later on but for now you need to know that it establishes starting values for G and D. G is set to half of N and D is set to N-G. First time though let's see if anything in the first half of the list can be swapped with its counterpart in the second half.

```
210 E = 0: REM E keeps track of any exchanges
220 FOR I = 1 TO D: REM look though part of the list
230 IFA$(I) = A$(I + G)GOTO280: compare the two
    parts
240 T$ = A$(I): REM temporary save
250 A$(I) = A$(I + G): REM swap
260 A$(I + G) = T$: REM complete
270 E = I: REM set the flag
280 NEXT: REM leaving off the I speeds things up a
    little
290 IF E THEN D = E - 1: GOTO 210
```

This line is the secret to this whole technique. E will equal the highest value of any swap or zero if there were none. Testing for E without specifying an equal is the fastest IF test. We've lowered the limit for the subsequent loop through the list and then we execute the loop. If E equals zero then we'll drop through to the next line of code.

```
300 G = INT (G/2)
```

This is the line that gives the shell sort its unique characteristics. Without the INT function a divide by two never gets to zero. The object is to parcel our list into smaller and smaller pieces until it's all done.

```
310 IF G THEN D = N - G: GOTO 210.
```

When G finally gets to zero there are no more pieces to parcel and we're done. The rest of the code simply proves that the list is indeed properly sorted.

```
350 REM . . . PRINT . . .
360 FOR X = 1 TO N
370 PRINT A$(X); "   ";
380 NEXT
390 DATA Q, W, E, R, T, Y, U, I, O, P, A, S, D, F, G,
    H, J, K, L, Z, X, C, V, B, N, M, END
400 END
```

Well, how did we do? In simplicity we were able to code a sort that only used one additional variable D. In addition we can still fit the code into five lines if we wish. Line 1 is BASIC line 200. Line 2 comprises lines 210, 220 & 230. Line 3 can be made up of Lines 240, 250, 260, & 270. Line 4 collects lines 280 and 290. And finally line five combines 300 and 310. The other criterion was speed and if we measure the speed of this sort we'll find it is very close to the quick sort. In our benchmark there is well under a tenth of a second's difference and if the list is partially ordered the new sort algorhythm wins by a wide margin! There are several things that you can still do to insure that the sort runs as quickly as possible. The standard rules apply for anything that needs to run fast.

1. Locate the routine near the beginning of your program to keep the time for GOTO's to a minimum.

2. Define the variables early in your program so that the variable search will take a short time. Note that even the order their specified affects the sort time.

3. Use multiple statements per line.

4. Leave out remarks.

5. Leave out the variable in next statements.

There are, however, some other things that you can do to make sure your sorts work as fast as possible. These things are things you may have overlooked about the state of your machine when programs are being run.

6. If you have a tool kit and it is enabled then you can expect your machine, and therefore your sorts, to slow down by 15%.

7. If you have loaded and run the wedge routine you can expect an additional 5% slowdown.

8. The interrupt routine that takes care of the clock and keyboard eats up another 10% of your machine. For the fastest possible sorts you can turn it off while sorting. POKE 59411,60 will turn it off while POKE 59411,61 will turn it back on. Be careful with this however as the stop key and the keyboard are disabled by this trick. Make sure that it is the last thing you do and there are no syntax errors encountered while the sort is going on. Also be sure and turn the interrupt back on when the sort is finished. There, that's all there is to it. I hope this will help keep your sorting time to a minimum . . . sort of.

TaxQwik thinks with you. TaxQwik is a tax preparation package* which analyzes your client's tax picture and recommends solutions. Take the option of filing separately or jointly. TaxQwik automatically calculates the most advantageous method and provides instantaneous income averaging comparisons. All in less than 5 minutes.

TaxQwik saves time every step of the way. The more complex the tax return, the more time you save. Especially when it comes to the time-consuming tasks of editing and checking. With TaxQwik, revised numbers are simply placed on the appropriate schedule and each section of the form is automatically retotaled. Potential time saved–70%. As for checking mathematical errors, forget it. There are none.

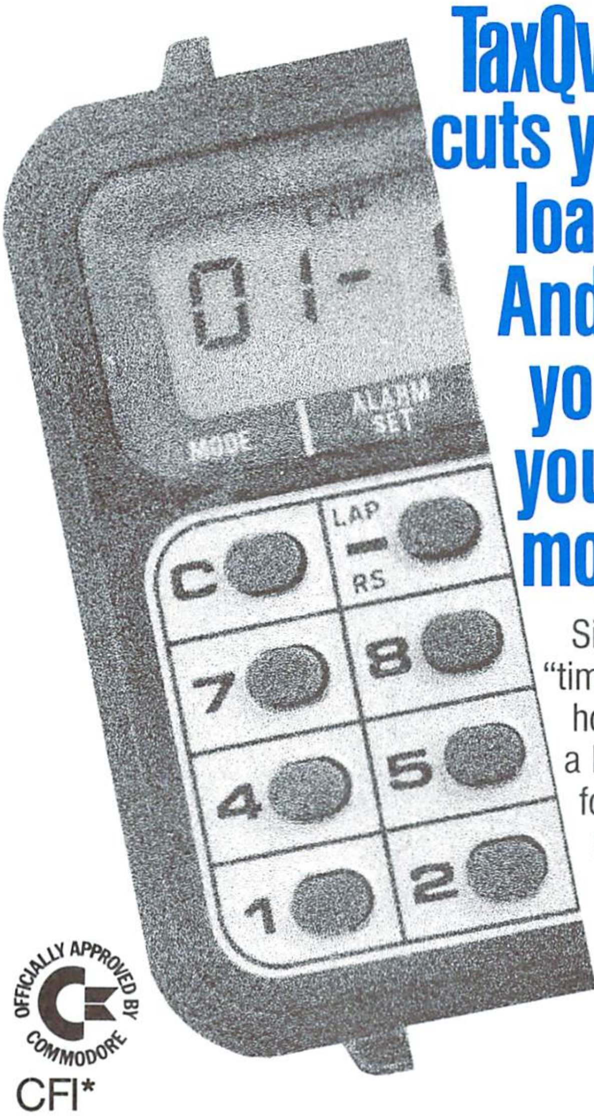
TaxQwik saves time by printing the form. TaxQwik automatically prints one page at a time on standard government tax forms. You eliminate expensive copying and collating. What's more, TaxQwik creates virtually all schedules. State modules are available.

Best of all, TaxQwik is fully supported and easy to use. When you buy TaxQwik you receive an easy to follow, step-by-step manual and a hotline which is open to you at all times. And we provide an annual revised program to keep up with changing tax laws.

TaxQwik is for you. Now. If you're big enough to have outgrown the outdated, laborious manual method for preparing tax returns, TaxQwik and the affordable micro-computer† is exactly what you need. Like other tax preparers, you will most likely double and triple your productivity. Moreover, during the year you'll find a dozen other ways to use your system. Like billing, data management, labels, correspondence and more. So don't waste another year. TaxQwik can pay for itself in one tax season.

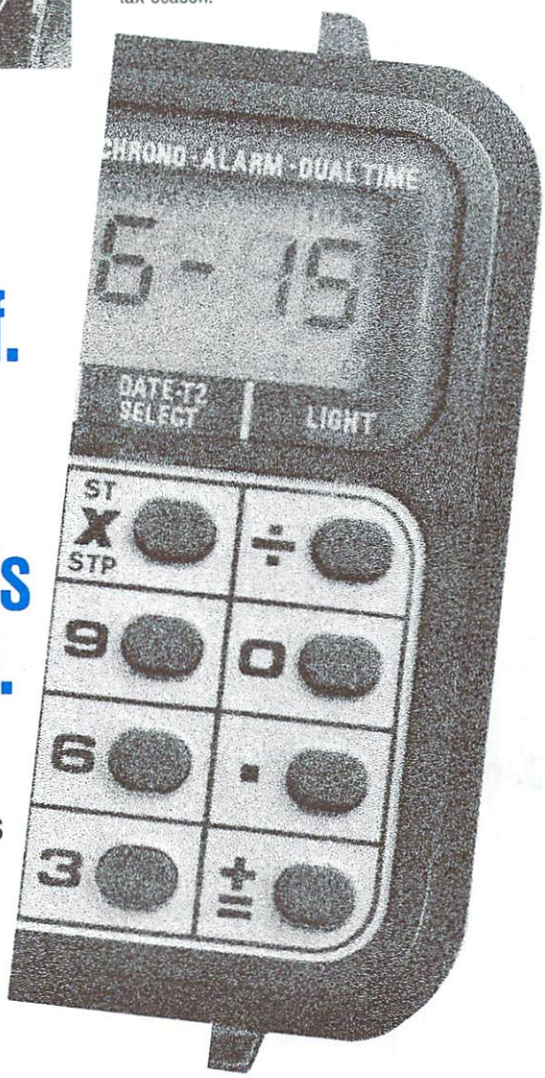
Call immediately: (201) 276-1144.
Ask for Leslie.

†Inquire about complete systems rentals for the tax season.



TaxQwik®
cuts your tax load in half. And helps you save your clients money too.

Since you sell "time" and "know-how", that means a lot more money for you – and a lot less pressure.



OFFICIALLY APPROVED BY COMMODORE

CFI*

*© COPYRIGHT 1982 CFI COMPUTER SOLUTIONS

TaxQwik saves time every step of the way.

GENEVA TECHNOLOGIES CORP.
14 Commerce Drive, Cranford, New Jersey 07016

Circle #17 on the Reader Service Card

6502 DECIMAL MODE

by Bruce Robinson

The 6502 has a convenient feature for machine language programming called the "decimal mode." When in this mode, addition and subtraction is done in BCD (Binary Coded Decimal). WARNING: INC and DEC instruction work as normal, only ADC and SBC operations are affected by the decimal mode.

What is Binary Coded Decimal? Instead of a byte representing a number from 0–255, it represents a number from 0–99. Each nibble (one half byte) represents a decimal digit (0–9).

EXAMPLES:

Binary	BCD
0001 0100	**0010 0001**
$15=21	**2 1**
0101 1110	**1001 1000**
$5E=94	**9 4**

Carries and borrows within the byte are taken care of automatically:

(Decimal mode flag = 1)

```
  0001 1001        19
+ 0000 0001   =   +1
  ---------       --
  0010 0000        20
```

The carry flag will be set if an addition exceeds 99, or cleared if a subtraction goes below 0.

(Decimal mode flag = 1)

```
  1001 0101        95
+ 0010 0010   =   +22
  ---------       ---
  0001 0111        17
```

Carry flag = 1

You still have to CLEAR (CLC) the carry before adding, and SET (SEC) the carry before subtracting.

A useful way to use the decimal mode is for keeping track of the score in a game. It is somewhat cumbersome to convert a binary score to an ASCII string for display (and very time consuming). By keeping track of the score in the decimal mode (i.e. represent and manipulate the score in BCD), conversion for display is much simpler.

The following program segment will convert a BCD score to ASCII for display:

```
        LDX#$(numberofbytes-1)
LABEL1  TXA
        ASL
        TAY
        INY
        LDA$ score, X
        AND#$0F
        CLC
        ADC#$30
        STA$ ascii, Y
        DEY
        LDA$ score, X
        LSR
        LSR
        LSR
        LSR
        CLC
        ADC#$30
        STA$ ascii, Y
        DEX
        BPL$ LABEL1
        RTS
```

The "number of bytes" is how many bytes you are using to represent the score.

The "score" is the address of the most significant byte of the BCD score.

The "ascii" is where in memory you want the converted BCD number to appear. You would most likely want this location to be on the screen.

Keeping track of things in BCD is useful whenever you have to communicate numbers to the person using the computer. Not only is it useful for the score, but also things like "pounds of fuel left," "altitude," "number of klingons left."

Multiple precision addition and subtraction is done the same way in the decimal mode as in the binary mode. Just remember to set the decimal mode (SED) before adding or subtracting BCD numbers, and be sure to clear the decimal mode (CLD) afterwards.

This program for PET and CBM computers is under development for the COMMODORE 64 as well. Should be available by spring.

PETSPEED Tips

by Joe Rotello, Jr.

With the introduction of the Petspeed™ BASIC Compiler, Commodore owners and programmers entered a new world of programming power. No longer were BASIC or machine language the only choices for PET programmers. Now, with the newest version of Petspeed (Petspeed 2.6), some of the little weaknesses in the older versions have been worked out, creating an even more exciting, easier-to-use, faster and more accurate product.

FOR . . . NEXT Statements
Compiled FOR . . . NEXT loops under Petspeed will perform far faster than they do in BASIC. For example, FOR I = 1 to 1000 will execute faster in its compiled form than it will in BASIC. So, for programs that use FOR . . . NEXT loops as timers, you will want to lengthen the time you assign to the index. For comparison, a few execution speeds are given in BASIC and Petspeed for sample values of "x"; the code used was "FOR I = 1 TO ×:NEXT". For the purposes of our test, the variable index "I" was left off of the NEXT command:

VALUE OF x	BASIC	PETSPEED
200	.25 sec	.05 sec
1000	1.08 sec	.23 sec
2500	2.75 sec	.58 sec
10000	11.71 sec	2.4 sec
25000	30.5 sec	6.0 sec

As can be seen, Petspeed gives the computer a real "kick in the ram" so to speak. Note also that all FOR . . . NEXT loops will execute faster under Petspeed, including those that are used to access the disk drive or drive the printer.

DATA Statements
Unlike BASIC, Petspeed allows string DATA statements up to 69 characters in length. Any more than that and the Petspeed compiler will crash with the error message that the DATA statement that was read from the BASIC source code was too long. Beware, as this problem will generally crop up when compiling already existing games or business programs where extensive use of lengthy DATA statements is made. Note that this information does not appear in the current Petspeed user manual.

POKES, PEEKS (and other creatures)
As it does for FOR . . . NEXT loops, Petspeed greatly speeds up the execution of POKE and PEEK statements. For example, the POKEing to the video screen of words or other characters that might have been too slow under BASIC will be speeded up by factors of 10 and greater. Some complex POKEs and PEEKs may benefit somewhat less from Petspeed than simple ones.

PRINT Statements
Many are dismayed to learn that Petspeed does not speed up PRINT statements (i.e., writing to the screen) any appreciable amount. At best, screen writing takes place perhaps two times faster than in BASIC.

GET$, GET# Statements
These statements are greatly speeded up, especially those that use the "get a single character and build up a string

from them" method. In fact, fetching bytes from the disk using GET# now becomes an efficient way to procure data when the program is compiled. If you enable the cursor when using GET to fetch from the keyboard, it will flash far faster, since the machine interrupt and cursor routines are running in high gear. No changes need be made here unless you might want to actually slow down the GET loops in the BASIC source code if the cursor is flashing too fast!

Formulas and Arguments
One item that may pop up on the Petspeed compiled program is "FORMULA TOO COMPLEX" errors. Usually these errors are caused by having too many continuous arguments for Petspeed to handle. In general, Petspeed will crash if there are three or more arguments but we have noticed that some "triple argument" codes will pass ok. Breaking up the BASIC code line into two distinct segments will not slow the compiler any noticeable degree, and Petspeed will greatly appreciate it.

For example, the following code will usually crash Petspeed:

```
50 a$(i) = a$(i)+mid$(a$(i)+right$(b$(i),len(a$)-1))
```

However, if we say:

```
50 j = len(a$):a$(i) = a$(i)+mid$(a$(i)+right$(b$(i),j-1))
```

Then Petspeed will swallow the code without any indigestion.

Sorts and More Sorts
All those users who love to write data sort routines into their BASIC programs but just could not stand the time it takes BASIC to perform even menial sorts are going to love Petspeed. In fact, Petspeed and sorts seem to have a natural affinity for each other. True, even a compiled sort routine won't win any speed races vs. true machine language, but try the following comparison for a change. We took a subscript sort in BASIC and used it to sort the disk directory, which the sorting performed in memory. In our test case, the directory had 64 entries and disk data was stored in a three dimensional array. In BASIC, the data took 41.5 seconds to sort. Using Petspeed, the identical sort took just under 2.4 seconds.

Of course, there are many types of sorts in existence. However, expect Petspeed to speedup most sorts by a factor of 14 to 20 times or more depending on the BASIC source code efficiency. Remember that BASIC sorts that move strings around will always be slower even under Petspeed. Try to use subscript sorts and other BASIC improvements to really get all that the compiler can give.

Logic Programs
As with all programs, there are some areas where the compiler will not be completely compatible with the BASIC source code. To exemplify, we ran a BASIC program that compares answers from a Dating Service questionnaire. The program consists mainly of repetitive logic comparisons and eventually the program arrives at the most perfect match it can. Under Version 2.5 of Petspeed the program logic had to be modified in order to be compatible with the compiler. However, Version 2.6 of Petspeed did not have this problem, so the BASIC source code did not have to be modified in any way. Although these problems are NOT widespread, SOME BASIC programs are not compatible with Petspeed 2.5 just as they stand. As in the case of logic situations, the Version 2.5 compiler operates under its own set of conditions. So, be sure that you have the latest version of Petspeed (2.6 as of this writing) and you will not run into these little quirks.

Using the Power Chip
Woe is the programmer who uses both the Power(TM) Programmers Utility Package chip and Petspeed without knowing of the strange and wonderful(?) events that may happen! Simply said, keying in "OFF" in order to disable the Power program before saving the BASIC source code to disk is NOT enough!

Save the completed BASIC source code to disk, and, if Power has or had been previously activated, call a system reset, say from the keyboard SYS 64790 (BASIC 4.0). Only then are you assured that Power and Petspeed will not attempt a merger that is doomed from the start.

If the system reset is not carried out, strange things like a locked disk channel, locked machine or a crashed Petspeed compilation will crop up at the worst times. (Remember, Murphy was a part-time programmer!)

Cassette Programs
Due to conflicts in address and cassette buffer locations, Petspeed is not suitable to use with cassette-based programs on the 8000 or 9000 series machine. Although this limitation may be eliminated in a later Petspeed release, compiled programs that attempt to load data or other programs from tape will fail, as the data transfer will not take place.

Version Improvements
Overall, differences between Versions 2.5 and 2.6 of Petspeed seem slight on the surface. However, deeper investigation reveals an on-going program of improvements. On the 8050 compatible program disk, the program entitled "Speed Up 8050" has been reduced in directory size from 36 blocks (2.5) to 1 block (2.6). The program now loads faster and has retained all the features of its predecessor.

As mentioned earlier in the article any BASIC source code incompatibility seen in Version 2.5 has been cleared up. Although the previous problems were not major by any measure, the improvement makes the program even more of a pleasure to use.

Math routines are slightly faster in Version 2.6, and the newer version now has the ability to manage three dimensional arrays, rather than just two dimensions in 2.5. A definite improvement for those code bangers who love (?) array work. By the way, for those who worry, programs compiled under Version 2.5 perform properly under Version 2.6.

We hope that the above tips help you find Petspeed easier and more enjoyable to use. C=

PET Relative File Reader

by Elizabeth Deal

One of the totally undocumented features of the DOS 2+ is the syntax used in working relative files in Upgrade (Basic 2) PETs. Fortunately, over a year ago Jim Butterfield reported (perhaps discovered, too) the magic words in the DOS. And, fortunately, the relative files themselves are the high point of the Commodore disk books, even though the books are silent on the Upgrade syntax. Putting the two together is not difficult, and indeed, we can work the files with little more effort than BASIC 4.

Here is a list of several references on the subject:

1. Jim Butterfield, "Mixing and Matching Commodore Disk Systems," *COMPUTE* #9, Feb. 1981.
2. Jim Butterfield, "Relative File Mechanics," *COMPUTE* #11, Apr. 1981.
3. Raeto West, *Programming the PET/CBM,* Level Ltd and COMPUTE Books Publication.
4. Karl Hilden, "The Relative Record File System," reprinted from *The TRANSACTOR* in *Commodore Magazine,* July 1981.
5. Sieg Deleu, "Using CBM 8032 and DOS2.0," *The TRANSACTOR,* vol. 3, #3, also in *Commodore Magazine,* Dec. 1981.
6. Your disk book explains just about all there is to know about those files.

Relative files are a splendid invention. They can double as sequential files, are fast, convenient, never need a rewrite and, of course provide random access. You can write or read any part of a record without string concatenation. This means that garbage collection should rarely be a problem.

Relative files can be coded in Upgrade PETs so long as we use the DOS' native language, rather than words such as RECORD in our programs.

Since BASIC 4 disk commands are apparently translated to the simpler commands the disk understands I see no reason for the enclosed program not to work on 4-systems. I have not used the side-sector or super-side-sector feature in calculating maximum records, hence, this should also work with all DOS systems that support relative files.

SOME BASICS

To read a file: ST = 64 means the end of record. Error #50 tells us we overshot the file in the record number request of channel 15. We must NOT try to read, as the system will probably crash in the worst case, and return bad ST in the best. The presence of 255 in the first position tells us a record is empty, this being more important in writing than reading.

Usually files contain "normal" data in ASCII form (name, address type of thing), in which case the only trouble can be in proper positioning for reading and writing within a record. This is done with carriage returns. At other times, a file may contain coded data, put out by print #f,chr$(code), in which case we may run into reading problems, since it is difficult to echo some characters to the screen because they are control characters.

FILE READER

And this is one reason for the attached file reading routine. The ultimate file reader, in terms of accuracy, is, of course, a disk dump, such as Jim Butterfield's DISK VIEW or DISK MOD program. The latter permits effortless experimenting in destruction of files. In fact, little would get done without those two. Big thanks to Jim for writing and sharing the utilities.

I wrote this reader just for looking at what is in the file, (you cannot change the contents) and trying to put some ▶

meaning into the values. In a way it is an interpretive program, but you will not see that part unless you do strange things or very wrong things. On normal, though perhaps, mismanaged records, you'll just see the normal-looking printout with several helpful indicators.

You may read the file in any order you wish. Numeric keypad keys are used to advance the record counter: 1 and 3 by plus/minus 1, 4 and 5 by 100, 7 and 9 by 1000. When a desired number is reached, push the RETURN key to read that record.

Alternately, to read sequentially, backwards or forwards push C key (continuous read) and hold on to the number key. To go back to random order push D (direct). Incidentally, in forward sequential reading we do not really need to do the RECORD-equivalent command in line 340, as the DOS takes care of positioning to the next record.

Every once in a while the counter gets stuck. Push any wrong key to fix that.

If you ask for a record with too high a number, PET will tell you about it. Eventually, if you hover around the area where you think the file ends, the PET will learn the number (line 710).

Q-key quits and closes the files. I recommend against using the STOP-key in any file handling programs.

Records get numbered, fields are on separate lines, all reading begins in byte one of the record. This organization is from PET's point of view. If the file is a mess, it may not reflect what you think is there or where it is, and this is also the reason for having an all-purpose reader. Please, bear in mind that the reader does not know about your logical file structure or any accessory index files.

CONVERSION OPTION

If you ask to see all characters, linefeed prints as reverse-j, carriage return prints as reverse-m and is acted on at the end of each field (chr$(13) and record (st = 64). Existing zeros in a record show up only if the end of record signal has not come in. Hence you may not see the trailing zeros. Squaring the record length with screen display will tell you what's happening. This is aided by a choice of making all spaces visible, as an apostrophe prints instead.

Other values may also show. They will be useful to people who in addition to writing normal ASCII files, put their information in coded numeric form by use of the CHR$(x) function. So long as x is a legal ASCII value, X will print normally. In the event of zero kids, thirteen bathrooms, and 147 sheep mowing the lawn, x needs to be handled differently, else the screen may clear on you in the best case.

Consequently, all such values print in the familiar PET manner: 147 comes out as reverse-shift-s, zero as reverse-@, etc. The advantage of the method is that the character count remains the same, but bear in mind that we are into interpretation. It is up to you to put the meaning into what you see. The point of the exercise is to make the record visible, and that it does. Once you know the record contains strange things, you can go after the exact contents with your own program.

MISCELLANEOUS NOTES and PROBLEMS

In DOS 2 and 2.5 relative files have side sectors (ss) containing 120 addresses. To arrive at the highest available record number (mx) take total blocks in file (bb) and record length (rl) from the directory and have the PET calculate records thusly:

(1) s1 = bb/120:s% = s1: ss = s% – (s1 <> s%), this tells us how many side sectors are in use. The rest are your records. So we compute:

(2) m1 = 254*(bb – ss)/rl:m% = m1:mx = m% – (m1 <> m%) – 1, where mx is the magic answer. Record count may be larger than you think as complete blocks are assigned to the file.

The 4022 printer does not print five characters correctly, hence the need for reversal in line 490. A quote sent to the printer as chr$(98) does not set the quote mode, but the screen still needs this handled by a poke in line 400. This is important since we introduce reverse characters.

Apart from such conversions, this is a straightforward GET#1,I$ . . . PRINT#2,I$ type program though it may not look it on first sight.

The 4022 acts silly in some situations. I wanted to do two things: detect whether it is turned on (line 190) and to switch it to lower case mode (line 280). Following the instructions in the book I coded both lines one after another. This didn't get me anywhere. When the printer was originally off, then turned on, the desired switch to lower case mode did not take place until I separated the coding in both time and space. Can anyone explain this strange event?

```
100 REM--------------------------------
110 REM RELATIVE FILE READER
120 REM ELIZABETH DEAL
130 REM--------------------------------
140 FORJ=0TO9:READAV(J):NEXT
150 FORJ=2TO7:READOF(J):NEXT
160 DV=8:RN=1:PS=1:SA=2:SV=96+SA:QQ=256:NT=255:CR=13:Z$=CHR$(0)
170 PK=0:PJ=2:DR=1:MX=65535:P=4
180 INPUT"PRINTER    N▮▮▮";I$:PRINT:D=3:IFASC(I$)=89THEND=P
190 CLOSE3:OPEN3,D,10:PRINT#3:IFST=-128THENPRINT"]* TURN IT ON *":GOTO190
200 INPUT"DRIVE    0▮▮▮";DR$:IFDR$<>"0"ANDDR$<>"1"GOTO200
210 INPUT"RELATIVE FILE NAME    *▮▮▮";FL$:PRINT
220 CLOSE15:OPEN15,DV,15:PRINT#15,"I"DR$:GOSUB690:IFEGOTO200
230 INPUT"BEGIN AT RECORD#    1▮▮▮";I$
240 Q=INT(VAL(I$)):IFQ<1ORQ>MXGOTO230
250 INPUT"SHOW SPACES (')    Y▮▮▮";I$:F=39:IFASC(I$)=78THENF=0
260 INPUT"SHOW ALL CHARACTERS    N▮▮▮";I$:AL=0:IFASC(I$)=89THENAL=1
270 RN=Q:CLOSE5:OPEN5,DV,SA,DR$+":"+FL$+",REL":GOSUB690:IFEGOTO200
280 IFD=PTHENPOKE59468,12:CLOSE3:OPEN3,D,7:PRINT#3:REM LOWER CASE PRINTER
290 CLOSE3:POKE59468,14:OPEN3,D:PRINT
300 IFALTHENPRINT#3,"▮ASC 0-31,128-159 --> REVERSE ASC+64"
310 PRINT#3
320 :
330 GOSUB560:IFEFGOTO440:REM QUITS
340 RH%=RN/QQ:PRINT#15,"P"CHR$(SV)CHR$(RN-QQ*RH%)CHR$(RH%)CHR$(PS)
350 GOSUB690:IFEGOTO440 :REM BAD
360 IFTBGOTO330 :REM TOO HIGH
370 IFD<>3THENPRINTRN
380 PRINT#3,"▮*"RN
390 GET#5,I$:SS=ST:IV=ASC(I$+Z$):IF(64ORST)<>64GOTO440:REM IF POSSIBLE
400 GOSUB460:PRINT#3,I$;:IFIQTHENPOKE205,0
410 IFSS=0GOTO390:REM KEEP IN RECORD
420 IFD=3THENIFDRTHENPRINT#3
430 PRINT#3:GOTO330:REM RECORD
440 PRINT#3:CLOSE3:CLOSE5:CLOSE15:END
450 :
460 UV=IVAND127:IR=(IV=CR)OR(SS=64):IFFTHENIFUV=32THENI$=CHR$(F):REM BLANKS
470 IQ=(IVAND63)=34:IFIQTHENI$=CHR$(98-128*(D=P)):REM QUOTE
480 IFAL=0THENRETURN
490 IFD=PTHENIFUV>90ANDUV<96THENI$=CHR$(IV-128*SGN(IV-128)):REM [\]↑←
500 IFUV<32THENI$="▮"+CHR$(IVOR64)+"▮"+CHR$(-CR*IR):REM WEIRD & CR
510 RETURN
520 REM---USER INPUT REC#------------
530 DATA 81,13,49,52,55,51,54,57,68,67
540 REM  Q  CR  1  4  7  3  6  9  D  C
550 DATA -1,-100,-1000,1,100,1000
560 IFDRTHENPRINT"]                ]":PRINTRN:REM KEEPS COUNTER IN PLACE
570 KY=PEEK(151):IFKY=PKTHENIFPK>0THENJ=PJ:GOTO610:REM SAME OLD KEY
580 GETI$:AV=ASC(I$+Z$):REM NEW KEY
590 FORJ=0TO9:IFAV<>AV(J)THENNEXTJ:PK=-1:GOTO570
600 PJ=J:PK=KY :REMEMBER KEYS
610 IFJ=0THENEF=1:RETURN
620 IFJ=1THENPK=-1:GOTO670
630 IFJ<8GOTO650
640 DR=J-9:GOTO560 :REM TRUE ON D
650 RR=RN:RN=RN+OF(J):IFRN<1ORRN>MXTHENRN=RR
660 IFDRGOTO560
670 RETURN
680 REM---FLOPPY STATUS--------------
690 TB=0:INPUT#15,E,E$,ET,ES:IFE=0THENRETURN:REM OK
700 IFE<>50THENPRINT:PRINT"*"E;E$;ET;ES:RETURN:REM FATAL
710 TB=1:E=0:PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮TOO HIGH":IFRN<=MXTHENMX=RN-1:REM EOF,NEW MAX
720 RETURN:----------------------------
```

LSI Exerciser

by George Kuetemeyer

One of the nice things about the 6502 family of microprocessors is the large number of available LSI support chips. These chips allow the experimenter or designer to easily add I/O lines, timers, sound effects, asynchronous communications, and high resolution graphics to home brew or commercially available systems. All of these support chips just hang on the 6502 bus. Consequently, they are very easy to add to any system. Of course, PET, VIC and Commodore 64 users are well aware of the advantages of these special function chips. Every Commodore product contains at least a few of these silicon marvels.

Most CBM machines, for example, use the 6522 VIA chip, a *very* versatile interface adapter. This 40-pin wonder contains two 8-bit I/O ports, two timers, a serial shift register, and special interrupt or "handshaking" capabilities. The VIC would not be a "VIC" if it were not for the exotic "Video Interface Chip" that provides the spectacular sounds and sights and I/O that we have come to expect from Commodore.

All of these chips implement hardware-type tasks, yet they can each be thought of and used as a series of memory locations. In fact, each chip contains a series of memory registers. Some of these registers will normally be written to. Some will read from. Others (such as I/O registers) will be used bi-directionally. Sometimes you will want to change one or more registers to accomplish a particular task. Sometimes you will want to only change one, two or three bits at a time.

You can experiment with these chips by peeking and poking with BASIC at the appropriate register addresses. (Be sure to keep your table of decimal-to-hex and hex-to-binary conversions nearby, you'll need it!!) Or, you can use the "LSI Exerciser". This program is set up to assist you in experimenting with these silicon gems. It was written for the 80-column CBM, but can be easily adapted to most other computers and screen configurations by simply modifying the "Set Up Screen" routine. You should also check the "Variables & Constants" routine and update the list for your particular configuration.

The program centers around a menu which allows selection of various operations. Generally, you should start by setting the number of registers. Most of these chips contain 16 registers, but there are some differences. You can then decide on labels to go with each register. You may enter them under program control or insert them in the dummy data statements provided. Once this is done, you should key in the chip base address. I have protected BASIC for PET/CBM users. VIC and Commodore users should change the "TB" (top of BASIC variable) in the variable list at the end of the program. These users will also have to modify line #130, which actually sets the top of BASIC. Check the memory map for your computer to change this.

You should also set up default values for each register (if you know what they are). This is especially important with video-type chips. You may find that your experimentation will lead to a display that will make you head for the nearest ophthamologist. When this happens, just hit the "@" key and everything should return to normal. You can enter the default values by modifying the dummy data statements provided after line 2250.

You are now ready to exercise your LSI chip. The display will show each register label and a list of initial values in binary, decimal and hexadecimal. You will find the binary representation handy as many registers are "control registers," with each bit having a different function. You can move the cursor around with the regular cursor controls. Move to the bit you want and key in either a "0" or "1". If you wish to update the register, press the "L" key. (I could have made the update automatic, but this approach allows you the luxury of correcting mistakes before they cause your machine to "run wild". This program, after all, assists you in poking around your system and, whenever you do this, you may find more than you bargained for).

If you press the "D" key, you can enter a register value in decimal, if this is more convenient. You will definitely want to use this feature when entering values into a pitch register in your VIC or 64. If you press the "H" key, you can enter values in hexadecimal. I have found this especially handy for doing hex conversions for other programs. The "<" and ">" keys are used to "scroll" the display backwards or forwards through memory or chip registers. This became a necessity with the introduction of the sound and video chips in the 64, as they contain more registers than could be displayed at one time on the screen. The scrolling is not fast, since the program is done in BASIC. You might try compiling the program with the PETSpeed compiler if you want to use it in a time-critical application. ▶

The program is set up in a modular form and can be easily adapted. If, for example, you wish to continuously observe data being received by the User Port, just create a loop which continually calls the "Read Registers" routine. If you want to add your own variables, don't forget to put them in the variable-list routine.

I've used the Exerciser in a variety of ways. One obvious case was an exploration of the S.I.D. (Sound Interface Device) chip in the Commodore 64. I used the decimal mode to set up pitch values. The binary mode was used to switch in various control register operations such as "synch" and "ring modulation" effects. Volume settings were done in hex since there are 16 possible sound levels available. I also investigated various properties of "sprite" graphics. In this case, however, I ran the exerciser on an 8032 and entered register values in the immediate mode on the 64. I did this to avoid destroying the Exerciser display. I've also often used the program to set up and check user port values.

The binary mode has been especially valuable, not only for exercising chips, but also for creating custom characters. Since each PET character is made up of eight consecutive data bytes, I simply set up a register display of eight bytes in memory. I can then enter "0"s and "1"s until I have exactly the image I need. I can then either record the hex values and enter them at a later date with the machine language monitor, or directly copy the contents of RAM into a 4K EPROM. The only step left is to put out my standard character generator and insert my own custom device. And, of course, whenever I need to work out hex, binary or decimal conversions, I always load in the Exerciser.

So go ahead, don't let your custom chips just sit around getting flabby. Put these new marvels of the computer age through their paces with the LSI EXERCISER!

```
READY.

 10 REM *****************************
 20 REM *
 30 REM *  LSI EXERCISER 10/20/82
 40 REM *
 50 REM *
 60 REM *   BY GEORGE KUETEMEYER
 70 REM *
 80 REM *
 90 REM *****************************
 100 REM *
 110 REM *   STARTUP
 120 REM *
 130 PRINT"{clr}";:POKE 52,224:POKE 53,46:REM SET TOP BASIC (12000 DECIMAL)
 135 GOSUB 2510:GOTO 160:REM INITIALIZE, GO TO PROGRAM START
 140 REM *
 150 REM *****************************
 160 REM *
 170 REM *   PROGRAM MENU
 180 REM *
 190 PRINT"{clr}";:REM CLEAR SCREEN
 200 GET K$:REM LOOK FOR INPUT
 210 PRINT"{down}{down}{down}{down}{down}{right}{right}{right}{right}{right}{right}{right}{rvs}1{off}  SET # CHIP REGISTERS"
 220 PRINT"{down}{down}{right}{right}{right}{right}{right}{right}{right}{rvs}2{off}  ENTER REGISTER LABELS"
 230 PRINT"{down}{down}{right}{right}{right}{right}{right}{right}{right}{rvs}3{off}  SET CHIP BASE ADDRESS"
 240 PRINT"{down}{down}{right}{right}{right}{right}{right}{right}{right}{rvs}4{off}  EXERCISE CHIP"
 250 IF K$="1" THEN GOSUB 320
 260 IF K$="2" THEN GOSUB 420
 270 IF K$="3" THEN GOSUB 740
 280 IF K$="4" THEN GOSUB 980
 290 PRINT"{home}";:GOTO 200:REM CONTINUE
 300 REM *
```

```
310 REM ******************************
320 REM *
330 REM *  SET # CHIP REGISTERS
340 REM *
350 PRINT"■■HOW MANY REGISTERS ARE THERE? (MAX=20)■":INPUTNR
360 IF AS=1 THENPRINT"RESTART PROGRAM":PRINT"■■";:RETURN
370 IF NR>20 OR NR<1 THEN GOTO 350
380 AS=1
390 PRINT"■■";:RETURN
400 REM *
410 REM ******************************
420 REM *
430 REM *  DEFINE REGISTER LABELS
440 REM
450 PRINT"■■DO YOU WANT TO INPUT A NEW REGISTER LABEL (NO=EXIT)■":INPUT KK$
460 IF MID$(KK$,1,1)="Y" THEN GOSUB 500:REM INPUT REG. LABEL
470 IF MID$(KK$,1,1)="N"THEN PRINT"■■";:RETURN:REM EXIT
480 IF MID$(KK$,1,1)<>"Y" AND MID$(KK$,1,1)<>"N" THEN GOTO 450:REM TRY AGAIN
490 REM *
500 REM ******************************
510 REM *
520 REM *  INPUT REGISTER LABEL
530 REM *
540 PRINT "■■INPUT REGISTER #":INPUT Y:IF Y<1 OR Y>20 THEN GOTO 540
550 PRINT"INPUT LABEL":INPUT RL$(Y-1)
560 KK$="":RETURN:REM EXIT
570 REM *
580 REM ******************************
590 REM *
600 REM *  DUMMY LABELS
610 REM *
620 DATA 1 XXXXXXXXXX,2 XXXXXXXXXX
630 DATA 3 XXXXXXXXXX,4 XXXXXXXXXX
640 DATA 5 XXXXXXXXXX,6 XXXXXXXXXX
650 DATA 7 XXXXXXXXXX,8 XXXXXXXXXX
660 DATA 9 XXXXXXXXXX,10 XXXXXXXXX
670 DATA 11 XXXXXXXXX,12 XXXXXXXXX
680 DATA 13 XXXXXXXXX,14 XXXXXXXXX
690 DATA 15 XXXXXXXXX,16 XXXXXXXXX
700 DATA 17 XXXXXXXXX,18 XXXXXXXXX
710 DATA 19 XXXXXXXXX,20 XXXXXXXXX
720 REM *
730 REM ******************************
740 REM *
750 REM *  SET BASE ADDRESS
760 REM *
770 PRINT"■■INPUT CHIP BASE ADDRESS"
780 INPUT BA:IF BA<TB OR BA >MB THEN GOTO 770
790 PRINT"■■";:RETURN
800 REM *
810 REM ******************************
820 REM *
830 REM *  SET UP SCREEN
840 REM *
850 PRINT"■■"
860 POKE 59468,12:REM GRAPHICS
870 POKE VO+Y*NC+X,CU:REM HOME CURSOR
880 PRINT"REG. LABEL     7 6 5 4 3 2 1 0
890 PRINT"■■■";
900 FOR I=0 TO NR-1:PRINT RL$(I):NEXT
910 FOR Y=0TONR-1:R$(Y)="0-0-0-0-0-0-0-0":RV(Y)=0:HX$(Y)="00":GOSUB 1940:NEXT
920 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■";
930 PRINT"■0■ OR ■1■ + CURSOR = BINARY, ■D■ECIMAL, ■H■EX, ■@■=DEFAULT,";
940 PRINT" ■R■EAD REG, ■<■ OR ■>■ = SCROLL, ■L■ = UPDATE REGISTER, ■E■XIT";
950 RETURN
```

```
960 REM *
970 REM ******************************
980 REM *
990 REM *  EXERCISE CHIP
1000 REM *
1010 GOSUB 820:REM SET UP SCREEN
1020 Y=0:UC=PEEK(VO+Y*NC+X)
1030 GETK$:POKE VO+Y*NC+X,UC:FORZ=1TOCD:NEXTZ
1040 PRINT"S                                        SBASE ADDRESS = ";BA
1050 IF K$=""GOTO 1190
1060 IF K$="Q"ANDY<NR-1THENY=Y+1:REM CURSOR DOWN
1070 IF K$="Q"ANDY>0THENY=Y-1:REM CURSOR UP
1080 IF K$="]"ANDX<28THENX=X+2:REM CURSOR RIGHT
1090 IF K$="]"ANDX>14THENX=X-2:REM CURSOR LEFT
1100 IF K$="0"ORK$="1" THEN GOSUB 1230:REM BINARY MODE: CHANGE BIT
1110 IF K$="D"THEN GOSUB1320:REM DECIMAL MODE
1120 IF K$="H"THEN GOSUB1430:REM HEXADECIMAL MODE
1130 IF K$="R"THEN GOSUB 2020:REM READ REGISTERS
1140 IF K$="@"THEN GOSUB 2130:REM SET DEFAULT VALUES
1150 IF K$="<" THEN GOSUB 2410:REM SCROLL BACK THROUGH MEMORY OR HARDWARE REG.
1160 IF K$=">" THEN GOSUB 2410:REM SCROLL FORWARD
1170 IF K$="L" THEN GOSUB 2340:REM LOAD VALUE  INTO MEMORY OR REGISTER
1180 IF K$="E" THEN PRINT"SQ";:RETURN:REM EXIT TO MAIN MENU
1190 UC=PEEK(VO+Y*NC+X):POKE VO+Y*NC+X,CU
1200 FORT=1TOCD:NEXT:REM CURSOR DELAY
1210 GOTO 1030:REM CONTINUE
1220 REM *
1230 REM ******************************
1240 REM *
1250 REM *  BINARY MODE
1260 REM *
1270 XX=X-13
1280 R$(Y)=LEFT$(R$(Y),XX-1)+K$+RIGHT$(R$(Y),LEN(R$(Y))-XX)
1290 GOSUB 1740:GOSUB 1650:GOSUB 1940:REM UPDATE DISPLAY
1300 RETURN
1310 REM *
1320 REM ******************************
1330 REM *
1340 REM *  DECIMAL MODE
1350 REM *
1360 FORKB=BBTOBE:POKEKB,32:NEXT
1370 PRINT"S";:PRINTSPC(30);:PRINT"     ▮▮▮▮▮";
1380 INPUT RV(Y):IFRV(Y)<0 OR RV(Y)>FF THEN GOTO 1370
1390 GOSUB 1820:GOSUB 1650:GOSUB 1940:REM UPDATE DISPLAY
1400 PRINT"S";:PRINTSPC(30);:PRINT"      ";
1410 RETURN
1420 REM *
1430 REM ******************************
1440 REM *
1450 REM *  HEXADECIMAL MODE
1460 REM *
1470 FOR B=BBTOBE:POKEB,32:NEXT
1480 PRINT"S";:PRINTSPC(30);:PRINT"     ▮▮▮▮▮";
1490 INPUT HX$(Y):IF LEN(HX$(Y))<>2 THEN GOTO 1480
1500 GOSUB 1540:REM CONVERT TO DECIMAL
1510 GOSUB 1820:GOSUB 1940:REM UPDATE DISPLAY
1520 RETURN
1530 REM *
1540 REM ******************************
1550 REM *
1560 REM *  HEX STRING -> DECIMAL
1570 REM *
1580 Z=FRE(0):HH$=MID$(HX$(Y),1,1):HL$=MID$(HX$(Y),2,1):REM HIGH, LOW NIBBLE
```

```
1590 FOR C=0TO15:IF MID$(HS$,C+1,1)=HH$ THEN HC=16*C:GOTO 1610:REM HI NIBBLE
1600 NEXT
1610 FOR C=0TO15:IF MID$(HS$,C+1,1)=HL$ THEN HC=HC+C:GOTO 1630:REM LO NIBBLE
1620 NEXT
1630 RV(Y)=HC:RETURN
1640 REM *
1650 REM *****************************
1660 REM *
1670 REM *  DECIMAL -> HEX STRING
1680 REM *
1690 Z=FRE(0):HI=INT((RV(Y)/16)):HH$=MID$(HS$,HI+1,1):REM HIGH NIBBLE
1700 HC=RV(Y)-(HI*16):HL$=MID$(HS$,HC+1,1):REM LOW NIBBLE
1710 HX$(Y)=HH$+HL$:REM COMBINE NIBBLES
1720 RETURN
1730 REM *
1740 REM *****************************
1750 REM *
1760 REM *  BINARY STRING -> DECIMAL
1770 REM *
1780 Z=FRE(0):DR$=R$(Y):BC=0:EX=7
1790 FOR C=1 TO 16 STEP 2:IF MID$(DR$,C,1)="1" THEN BC=BC+2↑EX
1800 EX=EX-1:NEXT:RV(Y)=BC:RETURN
1810 REM *
1820 REM *****************************
1830 REM *
1840 REM *  DECIMAL -> BINARY STRING
1850 REM *
1860 REM *
1870 Z=FRE(0):EX=7:DR$=R$(Y):BC=RV(Y)
1880 FOR C=1TO16STEP2
1890 IF BC<2↑EX THEN DR$=LEFT$(DR$,C-1)+"0"+RIGHT$(DR$,LEN(DR$)-C):GOTO1910
1900 IFBC>=2↑EXTHEN DR$=LEFT$(DR$,C-1)+"1"+RIGHT$(DR$,LEN(DR$)-C):BC=BC-2↑EX
1910 EX=EX-1
1920 NEXT:R$(Y)=DR$:RETURN
1930 REM *
1940 REM *****************************
1950 REM *
1960 REM *  UPDATE DISPLAY ROW
1970 REM *
1980 PRINT"SQ";:FORQ=0TOY:PRINT"Q";:NEXTQ:PRINTSPC(14);:PRINTR$(Y);"          ";
1990 PRINT"██████████";RV(Y);HX$(Y)
2000 RETURN
2010 REM *
2020 REM *****************************
2030 REM *
2040 REM *  READ REGISTERS
2050 REM *
2060 FOR Y=0 TO NR-1
2070 RV(Y)=PEEK (BA+Y)
2080 GOSUB 1820:GOSUB 1650:GOSUB 1940:REM UPDATE BYTE STRING
2090 NEXT Y:Y=0
2100 RETURN
2110 REM *
2120 REM *****************************
2130 REM *
2140 REM *  SET DEFAULT VALUES
2150 REM *
2160 FOR Y=0 TO NR-1
2170 RV(Y)=DF(Y):REM GET DATA
2180 GOSUB 2350:REM CHANGE REGISTER
2190 GOSUB 1820:GOSUB 1650:GOSUB 1940:REM UPDATE BYTE STRING
2200 NEXT Y:Y=0
2210 RETURN
2220 REM *
2230 REM *****************************
```

```
2240 REM *
2250 REM *  DUMMY REGISTER VALUES
2260 REM *
2270 DATA 000,000,000,000
2280 DATA 000,000,000,000
2290 DATA 000,000,000,000
2300 DATA 000,000,000,000
2310 DATA 000,000,000,000
2320 DATA 000,000,000,000
2330 REM *
2340 REM ****************************
2350 REM *
2360 REM *  LOAD CHIP REGISTER
2370 REM *
2380 POKE BA+Y,RV(Y)
2390 RETURN
2400 REM *
2410 REM ******************************
2420 REM *
2430 REM *  SCROLL MEMORY/REGISTERS
2440 REM *
2450 IF K$="<" AND BA>TB+1 THEN BA=BA-1
2460 IF K$=">" AND BA<MB THEN BA=BA+1
2470 PRINT"S                                        SBASE ADDRESS = ";BA
2480 GOSUB 2020:REM GET VALUES, UPDATE DISPLAY
2490 RETURN
2500 REM *
2510 REM ****************************
2520 REM *
2530 REM *   VARIABLES & CONSTANTS
2540 REM *
2550 TB=12000:REM TOP OF BASIC
2590 DIM RL$(20):FOR I=0 TO 19:READ RL$(I):NEXT I:REM REGISTER LABELS
2600 DIM RV(20):REM REGISTER VALUES
2610 DIM DF(20):FOR I=0 TO 19:READ DF(I):NEXT I:REM DEFAULT VALUES
2620 DIM HX$(20):REM HEX REG. VALUES
2630 DIM R$(20):REM BINARY DISPLAY
2640 NR=1:REM NUMBER OF REGISTERS (MAX = 20)
2650 BA=15*4096:REM BASE ADDRESS (DEFAULT)
2660 EX=7:REM EXPONENT COUNTER
2670 BC=0:REM BINARY COUNTER
2680 BB=623:REM BEGIN KEYBOARD BUFFER
2690 BE=BB+9:REM END KEYBOARD BUFFER
2700 K$="":KK$="":REM KEYBOARD STORAGE
2710 HC=0:REM HEX COUNTER
2720 HI=0:REM HI NIBBLE INDEX
2730 HH$="":REM HIGH NIBBLE, HEX STRING
2740 HL$="":REM LOW NIBBLE, HEX STRING
2750 HS$="0123456789ABCDEF":REM HEX DECODE STRING
2760 X=14:REM HORIZ CURSOR POSITION
2770 Y=0:REM VERTICAL CURSOR POSITION
2780 X=14:REM HORIZ CURSOR POSITION
2790 XX=0:REM CURSOR OFFSET
2810 NC=80:REM # CBM COLUMNS
2820 VS=32768:REM VIDEO RAM ORIGIN
2830 VO=VS+2*NC:REM HOME FOR PROGRAM CURSOR
2840 UC=32:REM VALUE UNDER CURSOR
2850 CU=81:REM CURSOR DOT
2860 CD=75:REM CURSOR DELAY
2870 FF=255:REM HEX $FF
2880 MB=65535:REM MAX BYTES AVAILABLE
2890 Z=0:REM DUMMY VARIABLE
2900 RETURN
```

A Star is Born

by Ira Neal

Have you ever wondered how those television commercials for the VIC 20 are made? Well, here is a little inside information. When a television program such as a commercial is produced, each scene is recorded on video tape. As the scene is being recorded on tape a 'time code' track is also recorded. The 'time code' identifies each video frame as to time (hours, minute, second) and frame (there are thirty frames per second). Later, after all the program material is recorded, a copy of the tape is made.

Then the program is edited with the aid of a computerized video editor. Three VTRs are used. One will record the edited program and others will have the original tape and the duplicate copy. The editor controls the VTRs (video tape machines). An edit list is constructed that contains the start and stop 'time code' of each individual scene. The editor will then position one source tape to the beginning of the first scene and the other source tape to the beginning of the second scene. The first VTR will playback the scene while it is being recorded on the third VTR. When the end of the scene is reached, the second VTR will start playback of the second scene and VTR one is stopped. This process is done many times until all the scenes are placed on the third VTR.

Ok, how did you get those great VIC 20 pictures? That was simple. A cable was constructed to connect the video (video low) and the audio from the audio/video connector on the back of the VIC to the patch panel at the studio. The VIC was substituted for one of the cameras. The output from the VIC 20 was recorded directly onto the VTR.

However, the scenes that involved a VIC 20 and live action created another problem. Have you ever noticed when a picture of a television is shown in a movie or on a television show, it seems to roll or have black horizontal bars? This is because that television is not synchronized with the camera. In a television studio the cameras, special effects generators, etc., are all synchronized by a master signal or 'gen-lock'. To synchronize the VIC display, so it wouldn't "roll," a special video controller IC (VIC Chip) was made. This IC had a new input signal added: video reset. When the vertical sync pulse from the studio sync-generator is fed into this input the VIC Chip's scan position registers are reset and a new frame is started. The video is then fed into a 'Digital Time Base Corrector' which corrects any minute timing errors. This puts the VIC in synchronization with the rest of the studio.

Program to WordPro File Converter

by Neil Harris

Here is a handy little program for those of you using WordPro on a Commodore PET or CBM. This program will take any program and convert into a file that WordPro can use. All control characters and graphics will stay the same.

The conversion is a 2-step process. First, LOAD the original program. Type the following line:

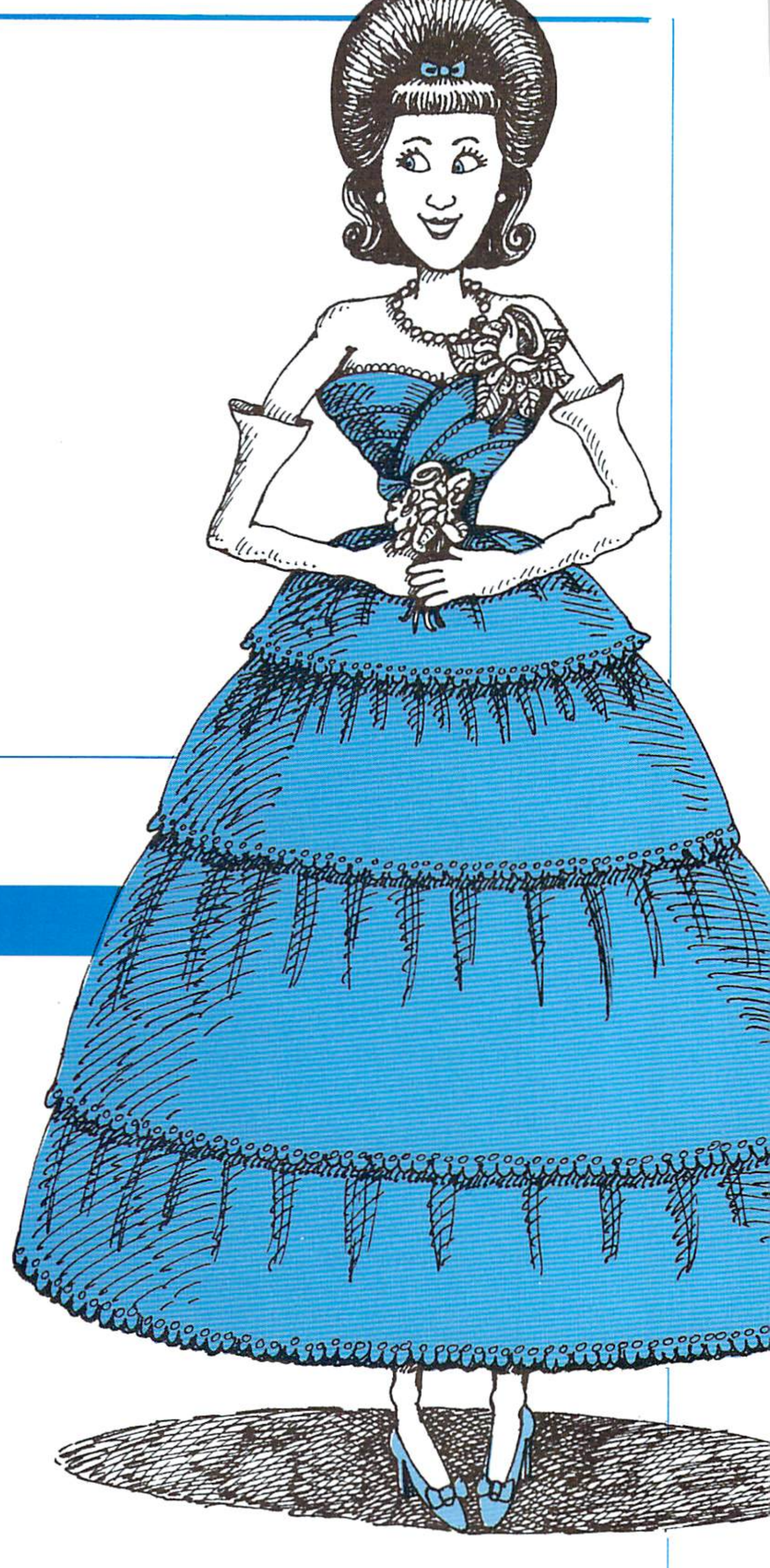
OPEN 1, 8, 2, " drive# : name ,P,W": CMD 1: LIST

The drive light will go on. When your cursor returns, type this line:

PRINT# 1: CLOSE 1

Now you have a temporary "listing" file that the program below can use. Just LOAD the converter program and type in the name of the "listing" file you just created and the name of the WordPro file you want to end up with.

```
10 REM PROGRAM TO WORDPRO CONVERTER
20 REM BY ---> NEIL HARRIS
30 INPUT "old file name: "; F1$
40 INPUT "file type (s,p)"; F3$
50 INPUT "drive#"; D1$: PRINT
60 INPUT "new file name"; F2$
70 INPUT "drive#"; D2$: PRINT
80 INPUT "number of columns for
    wordpro"; C
90 OPEN 1, 8, 2, D1$+ ":"+ F1$+
    ","+ F3$+ ",r
100 OPEN 2, 8, 3, "@"+ D2$+ ":"+
    F2$+ ",p,w
110 GET# 1, A$, A$
120 PRINT# 2, CHR$(0) CHR$(71);
130 GET# 1, A$: IF ST>0 THEN 260
140 IF A$= CHR$(13) THEN
    A$= CHR$(31): QM=0
150 IF QM=0 THEN 200
160 G= ASC(A$): IF G<32 THEN
    A$= CHR$(G+128): GOTO 210
170 IF G>63 THEN IF G<96 THEN
    A$= CHR$(G-64)
180 IF G>127 THEN IF G<161 THEN
    A$= CHR$(G+64)
190 IF G>191 THEN IF G<224 THEN
    A$= CHR$(G-128)
200 IF A$= CHR$(34) THEN QM= 1- QM
210 PRINT# 2, A$;: Q= Q+ 1
220 POKE 32768+ U,
    ASC( A$+ CHR$(0)): U= U+ 1
230 IF A$= CHR$(31) THEN IF Q<C
    THEN GOSUB 300
240 IF Q= C THEN Q=0
250 GOTO 130
260 CLOSE 1: CLOSE 2
270 OPEN 1, 8, 15, "s"+ D1$+ ":"+
    F1$: CLOSE 1
280 END
300 FOR Q= Q TO C-1: PRINT#
    2," ";: NEXT: RETURN
```



The Promqueen

reviewed by John Stockman

At long last there is a cartridge software development system for the VIC 20! The Gloucester Bus Company has designed a real winner here. With Promqueen, you can create your own machine-language programs on EPROMs (Electrically Programmable ROM's), programs like game cartridges and utilities.

The Promqueen is a single board cartridge-type peripheral which plugs into the VIC memory expansion slot. It looks like an oversized game cartridge and includes 4 mode select switches, 4 block select dip switches, 2 LED indicators, a reset button, 4K RAM, and a zero insertion force (ZIF) socket for 2K (2716) EPROMs, or 4K (2732 or 2732A) EPROMs. The Promqueen also comes with software on an EPROM called HEXKIT 1.0 and a fairly complete instruction manual. (note: Early Promqueens came with BASIC Hexkit, a BASIC program. Hexkit 1.0 is a new machine language program.)

It is important to read the instruction book before attempting to use the Promqueen because it may occupy any one of the four expansion blocks of the VIC 20 memory map and you must select the appropriate block, depending on where you want your machine code program to eventually execute.

After selecting the address space, inserting the unit and turning on the VIC, nothing happens because the Hexkit software does not auto-start. You must type in a SYS command to activate the software, which downloads into the VIC's RAM.

Hexkit and its utilities can be used to write machine code for a wide range of 8-bit microprocessors, like 6502, Z80, 8080, 8085, NSC800, and any others which generate 16 bit addresses in low-high order. The MIMIC mode allows the VIC to monitor the Promqueen RAM while an external computer is accessing that memory.

After Hexkit is activated, a menu of the 5 main options of Hexkit 1.0 is displayed:

EDIT HEX .... A
BURN EPROM .... B
SAVE ON DEVICE C
LOAD FROM DEV .... D
V'FY SAVE .... E

To begin simply press the letter of your choice on the keyboard. You can return from any option to the main menu by simply typing an 'X'. Hexkit itself can be stopped by typing 'X' at the main menu.

By far the most complex option from the menu is letter A, EDIT HEX. The option includes a wide number of useful commands for machine code programming. These commands include:

- **I** Initiate a machine code program.
- **J** Jumpback to display an address
- **K** Convert decimal number to hex
- **M** Assign one of 16 markers
- **N** Load the address of a marker
- **O** Find offsets
- **P** Position the display at a location
- **Q** Search for a 16 bit operand
- **R** Remap by adding offset
- **S** Stuff designated byte range with a value
- **T** Transfer a block of code
- **U** User defined operand search

In addition to these commands, the display scrolls bidirectionally by using the cursor up and down keys.

The Burn EPROM option of the menu is very simple, it is well prompted and all you have to do is answer each prompt and set the PROG/BURN switch to burn. The Promqueen will then burn, verify, and report any errors.

Machine code files can be loaded or saved from tape or disk. When saving, you must specify the range by either markers, decimal, or hex addresses. The other I/O options are much like those of VIC BASIC, but I/O errors are described by error number. In a case of a device not present, the error display would be 'I/O ERROR #5'.

The Promqueen CAN be used with VICMON, however there are certain restrictions to be aware of which are described in the Promqueen instructions.

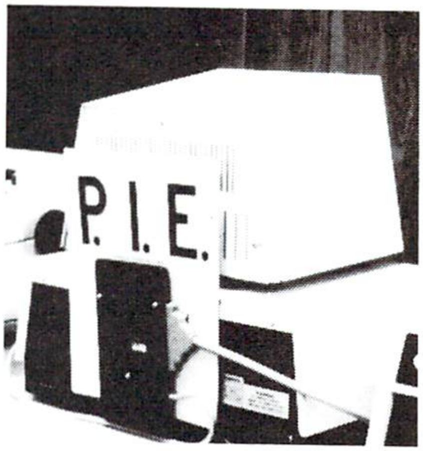
One of the most interesting features of the Promqueen is the ability to store BASIC programs on ROM. Yes, I said BASIC programs. The programmers at Gloucester Bus Company came up with a method for automatically uploading BASIC to the ROM, along with a short machine language program that makes the program auto-starting and auto-downloading. This means that your programs in BASIC run just like a game cartridge, without being obvious to the user that they are "just" BASIC.

The Promqueen is a well designed machine code cartridge software development system. I hope this device helps to expand the number of cartridges available for the VIC 20, because it surely makes this task much simpler. And the Promqueen makes the VIC into the least expensive development system available.

The Promqueen sells for $199, and is manufactured by Gloucester Bus Company, 6 Brooks Road, Gloucester, MA 01930. Call 617-283-7719.

Circle #27 on the Reader Service Card

PIE-C



PET/CBM* IEEE-488 TO PARALLEL PRINTERS By LemData Products

P.I.E.-C MEANS—Professional design, Indispensible features, Excellent quality and Cost effectiveness. You can't buy a better parallel interface for your PET/CBM.

Our P.I.E.-C will interface your PET/CBM through the IEEE-488 bus to ......

the NEC Spinwriter, the C. Itoh Starwriter, printers by Centronics, Epson, Anadex, Escon Products, the Paper Tigers by IDS, the MIPLOT by Watanabe, the DIP printers, the AJ-841, the OKIDATA printers, plus ALL OTHER parallel ASCII printers.

Assembled with custom case, CBM-TO-ASCII code converter and appropriate cable, the **P.I.E.-C** is only $119.95 (+$5 S&H). Md. Res. +5% tax. Specify printer and CBM models.

LemData Products, P.O. Box 1080, Columbia, Md. 21044 Phone (301) 730-3257
*PET/CBM are trademarks of Commodore Business Machines

Circle #28 on the Reader Service Card

DYNABYTE SOFTWARE™

By TSASA, INC.
IS
EXPLODING!!
WITH
BUSINESS AND
HOME SOFTWARE
For The

- COMMODORE 64
- VIC 20

Over 65 Cassettes Avail.
$8.95-$29.95

FREE CATALOG

DYNABYTE SOFTWARE
2 Chipley Run
West Berlin, N.J. 08091

CATALOG INQUIRY/REPORTING

for COMMODORE CBM computers

This fast and easy to use software is a must for micro computer users.

FEATURES:

- menu driven
- one step update
- wildcard inquiry and reporting
- hardcopy report generator
- unlocked basic programs
- enhanced DIRECTORY command
- many other features

SYSTEM REQUIREMENTS

- dual 8050 or 8040 disk drives
- 24k of memory

catalog no. r20v00 $29.95
specify 8050 or 4040 and CBM model

MSC software
1513 AVIATION BLVD.
SUITE A133
REDONDO BEACH, CA 90278

Circle #49 on the Reader Service Card

CBM is a trademark of Commodore Business Machines

Circle #29 on the Reader Service Card

VIC-20

VIC-20 INTERFACING BLUE BOOK

Did you know that your VIC can be used to control a 99¢ toy motor so effectively that it runs like a precision machine? Or that you can build an accurate digital thermometer using the VIC and four parts costing less than $5?

These and other 18 interfacing projects selected for usefulness, ease of construction and low cost are detailed in the VIC-20 Interfacing Blue Book, a veritable gold mine of practical information on how to build a variety of interfaces for your computer.

Projects include: Connecting VIC to your stereo; Pickproof digital lock; Capacitance meter; Liquid level sensor; Telephone dialer; Voice output; 8K/16K RAM/ROM expansion; 128K RAM expansion; 8-bit precision D/A; 8-bit A/D converter; MX-80 interface and more.

Written by a college professor in a friendly and informative style, the Blue Book gives you theory of operation, schematics, program listings, parts list, construction hints and sources of materials for each one of the 20 projects.

If you want to get the most out of your VIC this book is a must. Cost is $14.95 (less than 75¢ per project!). Price includes postage.

microsignal Dept C6
P.O. BOX 22
MILLWOOD NY 10546

Please send me a copy of the Blue Book.
Enclosed my check for $______

NAME ______________________

ADDRESS ______________________

____

____

Above prices include postage in the U.S. CA res. add 6% tax. Foreign add $2.

Circle #30 on the Reader Service Card

This latest book from Commodore is now available, only $19.95.

Review: Commodore 64 Programmer's Reference Guide

by Mike Heck

As a computer user, many times we take for granted the sophistication behind a system—like the Commodore 64—that allows it to obey our commands and perform intricate functions flawlessly. True, we may never have to go past learning how to use a particular application program like word processing; but at some point the curiosity within us naturally wants to understand more about what makes the system work.

While the Commodore 64 user guide reaches a new level in providing complete and easy-to-understand information, the **Commodore 64 Programmer's Reference Guide** fills in the gaps and provides more detailed technical descriptions, allowing you to unlock the full power of the 64.

The **Reference Guide** follows the format and concept of the **VIC 20 Programmer's Reference Guide** in that it has something for everyone, so people with various levels of understanding and interests will be able to benefit from it. For the casual user, you'll be able to explore more sprite graphics and music generation. System designers will find information on machine language programming, CP/M™ and other soft-loaded languages, plus interfacing various peripheral devices. And everyone can call on the BASIC section and handy appendices.

The **Reference Guide** is just that, a reference, and how much you get from it really depends on your prior knowledge of BASIC or 6502 machine language. However, the "semi-tutorial" approach of many of the sections, with technical jargon explained in plain English, helps you learn more about the 64 and put that knowledge to use, even if you are just starting out in computing. Each separate section of this 400-plus page guide covers a specific topic and can be used independently of the others. The topics covered include BASIC programming rules, BASIC language vocabulary, graphics, sound and music, machine language, input/output section, and detailed appendices.

The BASIC sections are similar to those in the VIC 20 guide. That's understandable since the 64 and VIC use the same Commodore 2.0 BASIC. Each command or statement is shown in the proper form with detailed examples of how it is used. The examples are really helpful as it is sometimes confusing figuring out how to write a command containing many options such as when I use the open command, what secondary channels need to be specified? The **Reference Guide** provides the answers.

In programming the goal is to design the most efficient way of doing things. The BASIC section shows how to abbreviate keywords and write programs that run quicker and use up less memory. One of the especially useful areas in this section is a description of the keyboard and the various ASCII codes generated each time a key is pressed

Graphics is one of the areas where the Commodore 64 shines, and the sprite/graphic section reflects it. This section even contains an elementary tutorial section for beginners on creating sprites and several graphic demo programs that you can modify to create your own sprite displays.

Other areas cover how to create programmable characters and your own special character sets. The tremendous amount of memory allows you to do some pretty amazing things with the 64.

For example, the **Reference Guide** explains how you can load a copy of the standard character set that is normally in permanent ROM memory into RAM (user memory), modify certain characters and then tell the system to use the new set. You also have a wide range of control over foreground, background and character color. How about blue characters with a yellow background on a white screen? It's all explained through simple one or two-line examples that can be typed in as you go along.

Bit mapping allows you to individually address each dot on the Commodore 64's screen to produce striking high-resolution graphics . . . in color. This information, along with a detailed explanation on creating and moving sprites—including worksheets to help you draw your own sprites—paves the way for creating sophisticated games, business graphics and . . . well, you decide.

Even if you rely on just one section, like music or graphics, it is worth the investment for that. And as was mentioned at the start, it is a reference piece. Some previous understanding is needed to completely grasp the complex subjects. Don't expect to be shown everything from the beginning. For that, a bibliography lists a number of good books to introduce you to a subject.

Music and sound synthesis is another ▸

area where the Commodore 64 has little competition. This section explains the various sections of the SID (Sound Interface Device) circuit and how sound is produced. Sound creation is perhaps the most difficult concept because there are so many variables. But even without a background in music theory you can get a lot out of this section.

Rather than starting with a technical description in this section, a short program is presented to introduce the concepts. Once accomplished, you are moved on to more complex tasks. For example, the Commodore 64 has four "voices" that can be used together to produce brilliant sound. But you are started using one voice and build up to more aggressive programs.

These additional programs show how to use different waveforms, the built-in filter, and such advanced techniques as syncronizing several voices and ring modulation to produce bell or gong sounds. All these complex concepts are illustrated, again, with clear examples that could be added into your own programs.

For simplicity, most examples presented throughout the guide are in BASIC, even though many would run much faster if written in machine code. In the machine language section, the reader is presented with a gentle introduction to machine language programming.

Since the 64 doesn't have a built-in monitor like the PET or CBM, the 64 MON machine language cartridge will be necessary to follow along with the examples. After a short intro and a guide to creating machine language programs, you are presented with 6510 microprocessor instructions, memory maps, using the KERNAL, and the likes, so most of this section isn't for novices.

As documentation goes, this section will probably get the most use, especially the memory map. It is invaluable for finding the correct memory location to PEEK, POKE, or work with in machine language to get a task done.

The last major section covers input and output—working with printers, disks, modems, and connecting other specific devices through the various ports.

For anyone serious about telecommunications or developing programs that use disk access, this section contains the needed information.

The appendices provide a quick reference for a variety of codes (display, ASCII, etc.), screen maps, music note values, pinouts and schematics, plus the specifications for a number of integrated circuit chips unique to the Commodore 64.

SUMMARY

While far from light reading, the **Commodore 64 Programmer's Reference Guide** contains valuable information for anyone using the 64, whatever your level or interests.

Commodore 64 Sprite Stuffer

by Elizabeth Deal

The logic of describing a sprite to the Commodore 64 computer is described on pages 69–71 in the user's guide. It involves a bit of arithmetic, and going through the process manually at least once is a necessary educational experience. Once that is done, however, it is time to turn the arithmetic over to the computer and leave the design to the user.

A sprite can be designed on the screen preprinted with line numbers, DATA and the quotes spanning exactly 24 positions. 21 lines fit well within a 25-line screen. A series of do-nothing dots in a REM statement is helpful for horizontal placement.

The following routine takes the picture(s) from DATA lines, does the necessary arithmetic and pokes the sprite map with the appropriate values. Each sprite begins on a memory address which is a multiple of 64. The first sprite goes at AD, which you should fill in. The remaining sprites follow contiguously and the 64-th byte is poked with a zero. You may wish to confirm this with the user's guide, as this code has only been tried on a PET.

There can be NS sprites. You may change NS to match the amount of DATA lines. The routine provides feedback by printing the sprite address and the picture, as well as three bytes to poke. You may, of course remove all PRINT statements when you think the code is all right.

Each sprite (I) has 21 SL$ DATA-lines (J). Each line (J) is made up of three groups (K) of eight characters (L). We poke three sums (SS) per line for a total of 63 bytes/sprite. SS is calculated by accumulating all non-space entries in the DATA lines (space is ASCII 32), using powers of two in the P2 array.

To prepare the screen with DATA lines, a code such as this does the job:

forj=1001to1021:printj"data" chr$(34)"[24 spaces]"chr$(34) :nextj

```
110 REM-----------------------------------
120 REM SPRITE STUFFER  ELIZABETH DEAL
130 REM-----------------------------------
140 FORIV=0TO7:P2(IV)=2↑(7-IV):NEXTIV
150 AD=*****:NS=1
160 :
170 FOR I=1 TO NS
180 PRINT:PRINT I;AD
190 FOR J=1 TO 21:READ SL$
200 PRINT:PRINT SL$;
210 FOR K=0 TO 2 :SS=0:SB=8*K+1
220 FOR L=0 TO 7
230 IF ASC(MID$(SL$,SB+L))-32 THEN SS=SS+P2(L)
240 NEXT L
250 PRINT SS;
260 POKE AD,SS:AD=AD+1:NEXT K,J
270 POKE AD,0:AD=AD+1:NEXT I
280 END
290 :
300 REM DESIGN SPRITES IN DATA LINES
1000 REM::.......................0001
1001 DATA"         XXXXXXX        "
1002 DATA"       XXXXXXXXXXX      "
1003 DATA"      XXXXXXXXXXXXX     "
1004 DATA"      XXXXX   XXXXX     "
1005 DATA"     XXXXX XXX  XXXX    "
1006 DATA"     XXXXX XXXXXXXXX    "
1007 DATA"     XXXXX XXX  XXXX    "
1008 DATA"      XXXXX   XXXXX     "
1009 DATA"      XXXXXXXXXXXXX     "
1010 DATA"      XXXXXXXXXXXXX     "
1011 DATA"      X XXXXXXXXX X     "
1012 DATA"       X XXXXXXX X      "
1013 DATA"       X  XXXXX  X      "
1014 DATA"        X  XXX  X       "
1015 DATA"        X  XXX  X       "
1016 DATA"         X  X  X        "
1017 DATA"         X  X  X        "
1018 DATA"          XXXXX         "
1019 DATA"          XXXXX         "
1020 DATA"          XXXXX         "
1021 DATA"           XXX          "
1100 REM::.......................0002
1101 DATA"                        "
1102 DATA"                        "
1103 :          SECOND SPRITE
1121 DATA"                        "
1200 REM::.......................0003
1201 DATA"                        "
1202 DATA"                        "
1203 :          THIRD SPRITE
1221 DATA"                        "
1300 REM::.......................0004
READY.
```

One Line Word Processor for the CBM 8032

By Keith Peterson

Sooner or later everyone needs a word processor. Some of us need one enough to spend the $400 for a really good one, or we buy a $40 one and get by. And most of us have tried a baby word processor at some time or another, but never got a really workable program.

This program is here to take you away from all that. It has a limitation or two, but is a workable system for word processing. And the price is definitely right!

The one line listed below does the whole thing. Just enter it as line 0 and enter your text as program lines behind it.

There are two things to remember about entering text. First, unless you begin each line with a quote or a colon the leading blank spaces will be eliminated, just like a normal program line. Secondly, if you don't use a quote for the first character, capital letters will not be printed.

This is a good time to use the auto numbering function of your favorite utility ROM. I usually just type a line containing the line number and a quote and duplicate it 100 times or so. (with a new line number each time, of course). Then I can just type in the text using the 8032's internal tab function to tab over to the first text column on the screen.

When the text file is printed, the last line will not be printed unless you enter the direct command 'print-#4,close4'. It's easier to just enter a blank line at the end of the file, and save the extra typing.

Once you have the text in a program file, you can use the screen editor to correct misspelled words or transfer lines of text, and you can use functions like renumber or find to locate words, just like the file is a regular program.

To print the text, just type run and touch return. The printer will spring to life and print out the letter, in upper and lower case, just like you typed it in.

If you wish to add screen output to the program, add this line:

```
1 poke19,32: list 2–
```

Enter run 1 and return to execute the output to screen.

When the text has been printed, either to screen or to printer, you must enter a character or two of garbage and touch return. I always enter a 'k' (for kill). This will generate a syntax error and restore the cursor to its normal operation. Crude but effective.

In certain situations, you may need to enter poke19,22 directly to get the line numbers back.

So what are the constraints, you ask? Just one, really. Whenever you print the text, there appears a column of quotes in the first column. There are two quick and dirty ways to solve this problem.

The first is to position the paper in the printer so that the quote always prints on the removable left tractor feed margin of your paper. The text margins will have to be suitably set when you enter the text, but the quotes will be torn off and forgotten.

The second way is really cheap and dirty. Just slide a scrap of paper into the printer right where the quote is printed. The printer prints the quote on the scrap instead of printing it on the letter.

The only other problem that you will have is the lack of utility functions like paging, page numbers, automatic margins, justification and so on. If you need these things, then by all means buy a word processor.

But if you type just a few letters, this one liner allows you to type out text just how you want it, and lets you save the complete file, text, program and all, on disk or on tape, as quickly as any word processor.

So you can type whatever you wish, even graphics, save it on disk or cassette, and print it on screen or printer. What more could you ask for one line of code?

References: Lindsey Doyle in Printout, **Feb 81 and Andrew Stezsyn in** The Paper, **Vol III, issue 8/9/10**

Program listing:

```
0 poke19,32:open6,4,7:print#6:close6:open4,4:cmd4:list2-
2 :
3 "                    (enter text here)
4 "   Dear Sir,
5 "        This letter is to inquire into my need for your word
6 "   processor. Why should I spend $400 for your program when
7 "   I typed this out to you for free?  It is my opinion that
8 "   this is all the word processor I really need.  Thank you
9 "   for your interest, but I'll keep the cash.
```

And so on.....

KP

Automatic Program Line Generation on the 8032

By Keith Peterson

There have been a wealth of programs written to save time and effort when writing programs. This program adds one more tool to the programmer's list of program-writing utilities.

This program automatically generates program lines with repetitive entries. For example, you can create 100 data lines, ready to just type in the numbers. Or if you want to enter a lot of remarks, you can create 50 remark lines, and write in the text afterwards. You could even generate remarks with a quote after them to suppress the tokenization of capital letters in 4.0 basic.

But the best use for this program is in conjunction with the one-line word processor described earlier. The text in that program is written on program lines containing only a quote. This program will create as many program lines as you wish, each with one quote, saving you the task of building up all those line numbers every time.

Two program listings are given to show two possibilities. The first allows user input of the data to be placed on each line. The second defines that data within the program.

Start with the first listing. Type in the five lines at the beginning of the program. The END statement is important: don't leave it out!

When the program is entered, type RUN. The program will prompt you for data. This is the data that will be entered on each of the lines that will be created.

Line four increments the line number, prints the current line number in two places, and executes the HOME command. It then tests for the last line number and clears the screen if it has been reached.

In either case, line five ends the program. At this time the cursor is in the home position, so a READY is printed and the computer goes about emptying the keyboard buffer. The first character in the buffer is a return, so a return is executed. Since the cursor is sitting on the line that was printed by the program, that line is executed, and the line is stored.

The next return resets the current line number to the value it had before the storing of the line above eased it. The third return resets the 'last line to enter' variable, and the fourth return executes the GOTO3. Whereupon the process repeats.

When the final line number has been reached, the program will print a clear screen. Now the returns pushed into the buffer in line 3 have no commands to execute, and come out as just line feeds. The program thus stops.

If you wish to use different line numbers for this routine, no harm will be done. Just be sure to change the GOTO3 in line 2 to reference the new line number that the current line three is given.

If you wish to increment the line numbers by two, by ten or by any number, just change the statement in line three. For example, for lines incremented by eight, change n=n+1 in line three to n=n+8.

With a little imagination you can go beyond this program to create self-writing and self-modifying programs that do all sorts of marvelous things. But in the meantime, this simple application can take a lot of the drudgery out of programming.

This data input, since it uses the input command, is limited in the characters it will accept. Commas, semicolons and quotes should not be used. The second program listing shows the use of these characters: we'll get to that in a moment.

After entering the data you wish to have placed on each line, the program asks for the starting line number and the ending line number. These should be chosen with the realization that any existing line with line numbers in that range will be deleted.

Once the line range has been entered, the screen clears and the program takes off. It will let you know when it's done.

When it finishes, list the program. The data that you entered is seen in each line in the range that you specified.

The second program listing is just like the first listing with the exception of the first line. This new first line defines the data directly so that you can enter data that would not ordinarily be accepted by the input statement.

To define the data, just enter a statement setting the string DA$ equal to the string that you would like to use. This program takes that yet a step further, showing how to handle the extreme case of the quote.

The quote is a special case because of the way that it affects cursor controls. One solution is to print an escape after the quote. But that wouldn't help those of us who don't have an 8032. So this program prints a quote, deletes it, and prints it again. Thus an even number of quotes were printed, yet only one is displayed.

If you have unusual data that you wish to enter on several lines, you can use similar logic to modify the program to your specific needs.

Let's take a look at how the program works. It uses what has been referred to as the dynamic screen to create these line numbers. Certain commands are placed on the screen, four returns are pushed into the keyboard buffer, and the program is exited. The returns execute the commands, and the process starts over.

In line one that data is set and the beginning and ending line numbers are entered. As discussed above, this line may vary depending on your applications.

Line two prints the commands that will do the work on the screen. This includes the data to be entered, the ending line number and a statement to GOTO3. The restating of the variables is necessary since any additions or modifications to a program erase all the variable values. The GOTO3 then allows the program to continue.

Line three puts four returns into the keyboard buffer, and sets the keystroke counter to four.

Line four increments the line number, prints the current line number in two places, and executes the HOME command. It then tests for the last line number and clears the screen if it has been reached.

In either case, line five ends the program. At this time the cursor is in the home position, so a READY is printed and the computer goes about emptying the keyboard buffer. The first character in the buffer is a return, so a return is executed. Since the cursor is sitting on the line that was printed by the program, that line is executed, and the line is stored.

The next return resets the current line number to the value it had before the storing of the line above erased it. The third return resets the 'last line to enter' variable, and the fourth return executes the GOTO3. Whereupon the process repeats.

When the final line number has been reached, the program will print a clear screen. Now the returns pushed into the buffer in line 3 have no commands to execute, and come out as just line feeds. The program thus stops.

If you wish to use different line numbers for this routine, no harm will be done. Just be sure to change the GOTO3 in line 2 to reference the new line number that the current line three is given.

If you wish to increment the line numbers by two, by ten or by any number, just change the statement in line three. For example, for lines incremented by eight, change n = n + 1 in line three to n = n + 8.

With a little imagination you can go beyond this program to create self-writing and self-modifying programs that do all sorts of marvelous things. But in the meantime, this simple application can take a lot of the drudgery out of programming. C=

```
 1 input"{clr}line data";da$:input"{down}{down}starting line #";ns:input"{down}{down}ending line #";ne
 2 print"{clr}{down}{down}{down}"spc(6)da$"{home}{down}{down}{down}{down}n={down}{down}{down}ne="ne"{down}{down}{down}goto3":n=ns
 3 forx=623to626:pokex,13:next:poke158,4
 4 n=n+1:print"{home}{down}{down}{down}"n"{down}"n:print"{home}":ifn=nethenprint"{clr}all done"
 5 end
ready.
```

```
 1 q$=chr$(34):da$=q$+chr$(20)+q$:input"{down}{down}start line #";ns:input"{down}{down}end line
   #";ne
 2 print"{clr}{down}{down}{down}"spc(6)da$"{home}{down}{down}{down}{down}n={down}{down}{down}ne="ne"{down}{down}{down}goto3":n=ns
 3 forx=623to626:pokex,13:next:poke158,4
 4 n=n+1:print"{home}{down}{down}{down}"n"{down}"n:print"{home}":ifn=nethenprint"{clr}all done"
 5 end
   ready.
```

Microphys Programs

PET

PET

2048 Ford Street
Brooklyn, New York 11229
(212) 646-0140

VIC-20

VIC-20

FOREIGN AND DOMESTIC DISTRIBUTOR INQUIRIES WELCOMED.

Microphys, a leader in educational software development, is pleased to announce the release of several recreational software programs for use with the Commodore VIC-20 microcomputer. The VIC programs, described below, require a 3-K expansion cartridge and utilize the VIC's excellent color graphics and sound capabilities. Each program retails for $15 and is accompanied by complete instructions.

PROGRAM DESCRIPTIONS

PV901 - Missile Math: this program presents in a game format, an opportunity for youngsters (ages 5-15) to practice and develop the basic skills of addition, subtraction, multiplication, and division. Four levels of difficulty in each skill area may be selected. Problems in a given skill are randomly generated and missiles are launched at correct answers. The computer displays the results on each program run and may be directed to generate the same sequence of problems so that review and 'match play' (against an opponent) are possible.

PV350 - Cryptograms: this program permits the generation of 'secret' messages which are to be decoded. These cryptograms are displayed along with their unique code number classifications. To decode a cryptogram, the program is run from line 9000. Family members can challenge each other with their individually created messages. If you enjoy solving the cryptograms appearing in newspapers and crossword puzzle magazines, this program is perfect for you. Note: two VIC users may exchange encoded messages. User 1 creates a secret message and transmits this to user 2. The code number will permit user 2 to have **his** VIC decipher the message should he encounter any difficulty.

PV340-349 - Anagrams: this series of programs provides an educational challenge for virtually all age groups. The VIC randomly generates scrambled words which are to be identified. Two clues are provided in order to assist in this process. The clues in the school and college categories are generally definitive in nature. Many of the words used are part of the Microphys Spelling and Vocabulary series for the associated grade levels. Thus, reading, vocabulary, and spelling skills are reinforced by these Anagram programs. Note: the same sequence of words generated may be requested so that 'match play' is possible. There are 5 level-of-difficulty categories each consisting of two programs.

PV340-341 Recreational	PV342-343 College	PV344-345 High School
PV346-347 Junior High	PV348-349 Elementary	

PV375-380 - Wheel-of-Fortune Word Games: this series of programs represents an exciting challenge for every member of the family. Players try to fill in missing letters in a randomly generated title or phrase and earn and lose points according to the graphic display on a 'Wheel-of-Fortune'. The scores of as many as four players are displayed, 1000 points being required to win a given game.

PV375 Song Titles	PV376 Famous Places	PV377 Entertainers
PV378 Statesmen	PV379 Scientists	PV380 Sports Figures

PV601-644 - Missile Spelling: this series of 36 programs enables youngsters in grades 4 through 12 to practice and develop basic spelling skills. Each program contains 60 graded words. The VIC randomly selects groups of 5 words, one of which is spelled incorrectly. Missiles are launched in order to destroy the word misspelled. The words chosen for grades 7 - 12 correspond to the Microphys Vocabulary series. Note: there are 4 programs in each grade level.

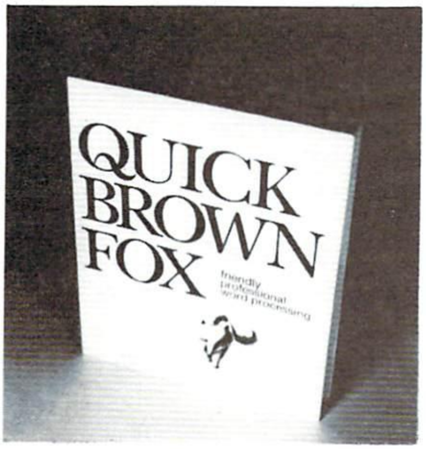
PV601-604 Grade 12	PV606-609 Grade 11	PV611-614 Grade 10
PV616-619 Grade 9	PV621-624 Grade 8	PV626-629 Grade 7
PV631-634 Grade 6	PV636-639 Grade 5	PV641-644 Grade 4

PV401-460 - Vocabulary: each vocabulary program randomly generates graded words which are to be defined. A sentence, in which the word is properly used, is displayed when an incorrect response is made. Using this contextual clue, a second opportunity to define the word is given. Reading and spelling skills are also reinforced as a more powerful vocabulary is developed. There are 10 programs in each grade level.

PV401-405 and PV431-435 Grade 12	PV406-410 and PV436-440 Grade 11
PV411-415 and PV441-445 Grade 10	PV416-420 and PV446-450 Grade 9
PV421-425 and PV451-455 Grade 8	PV426-430 and PV456-460 Grade 7

Educators should write for the new Microphys Fall Catalog which describes over 200 programs for use in Chemistry, Physics, Calculus, Mathematics, Vocabulary, and Spelling classes on both the high school and college levels.

NEW PRODUCT DEVELOPMENTS



Company:

Quick Brown Fox
548 Broadway, Suite 4F
New York, NY 10012
212-925-8290

Product:

Quick Brown Fox—Word-processor for the VIC 20 and Commodore 64. Allows full line and global editing, text moving, boilerplating, tab and margin settings, right justification and proportional spacing, even with the VIC 20 22-column screen width. Automatically reformats edited text.

Cost:

$65.00

Company:

MAG, Inc.
493 East Clayton Street
P.O. Box 346
Athens, GA 30603

Product:

The Contractor—A job costing system as well as a general accounting program for the CBM 8032 and 8050 disk drive. It is fully integrated and interactive, requiring only one data disk and one program disk. The program will handle up to 200 jobs at one time, as well as General Ledger, Accounts Receivable, Payroll and Inventory.

Cost:

Contact company

Product:

MAGIS Plus—A management information system designed to run on the CBM 8032 with 8050 disk drive. MAGIS Plus is an entire, fully automated accounting system designed to be used by personnel without special knowledge of accounting or microcomputers. It is a fully integrated and interactive system requiring only one data disk and one program disk. It is menu driven, with screen prompts. The system then automatically posts all information to the affected accounts.

Cost:

Contact company

Company:

Specialised Software, Inc.
P.O. Box 2
Wilmot, WI 53192
414-862-6968

Software for Commodore computers, each with manual, operating instructions and worked through examples:

Product:

BEP, Single or Multiple—Finds the financial Break Even Point for a single or multiple product line from cost and volume data.

Cost:

$40.00

Product:

AMORTIZE—Develops an amortization schedule for time-based installment purchases.

Cost:

$22.50

Product:

SECURITIES—Finds real earning per share under plowback, stock retirement, mergers and acquisitions, leverage, increased sales and stock dilution.

Cost:

$75.00

Product:

CATHNIP-Finds impact on delivery schedule of variations in failure rate of required building blocks. (This program does all scheduling, with PERT as a subset)

Cost:

$500.00

Product:

FOURIER ANALYSIS & FOURIER SYNTHESIS—Given point pairs, find Fourier Equations for interpolation; given coefficients and arguments find the value of the function.

Cost:

$50.00

Company:

Cacti Computer Services
130 9th St., S.W.
Portage la Prairie, Manitoba R1N 2N4
Canada

Product:

Large Pressure Sensitive Keyboard for PET/CBM—For physically handicapped users. 11″ × 21″ pressure sensitive keyboard with widely spaced key contacts to enable users with limited finger and hand control to operate a microcomputer. Several keyboard layouts are available, and custom arrangements can be made at no extra charge. Includes interface, connecting cables, plastic mask and driver routine.

Cost:

$525.00 U.S. currency

NEW PRODUCT DEVELOPMENTS

Company:
RAK Electronics
P.O. Box 1585
Orange Park, FL 32073

Product:
C64 FILE—A multi-purpose data base management system for the Commodore 64. Allows user to construct, sort, maintain and print out a relatively wide range of data types. Suitable for books, records, accounts, and mailing lists. User can load an existing file from cassette or start a new file with the command LOAD. Other function commands include DUMP, PRINT, ADD, CHANGE, REMOVE, SORT and QUIT.

Cost:
$9.95 plus $2.00 shipping and handling

Product:
FILEWRITER/FILEREADER—For the Commodore 64. Allows you to produce data files on cassette tape from information you type in, and allows you to read any data files stored on cassette tape. Offers output to screen or printer.

Cost:
$6.95 plus $2.00 shipping and handling

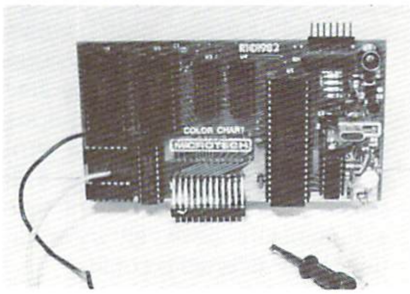
Company:
Dilithium Press
11000 S.W. 11th Street, Suite E
Beaverton, OR 97005
503-646-2713

Product:
Small Business Computer Primer by Robert B. McCaleb—A step-by-step guide for businessmen planning to buy a computer system, written in straightforward business language. Shows the abilities and limitations of the computer, includes worksheets, case studies, a reference section, glossary and shopping systems lists.

Cost:
$12.95 paper

Product:
Microsoft BASIC, 2nd Edition by Ken Knecht—Completely updated version, starts the reader off with an introduction to programming in BASIC and a glossary of computer terms. Covers topics such as branching and loops, arithmetic in BASIC, strings, editing, arrays and files, the disk and other useful features. Requires only a basic understanding of computer fundamentals.

Cost:
$14.95 paper



Company:
CGRS Microtech
P.O. Box 102
Langhorne, PA 19047
215-757-0284

Product:
Color Chart—Color video RAM board for PET/CBM. The small $2^1/_2''$ by $5''$ board plugs into a 2532 ROM socket. Two control wires clip to read/write signals in the system and convert the ROM socket into a 4K video RAM. Can be used to present independent color graphics displays on a color video monitor or TV (with RF modulator) while the main screen displays corresponding text. Operates in eight different modes, ranging from an alphanumeric 32×16 display with built-in character generator to a high-resolution graphic mode with 128×196 pixels. Up to eight different colors are available, depending on the mode.

Cost:
$139.95

Company:
Distribution Unltd.
P.O. Box 81702
San Diego, CA 92138
714-299-3718

Product:
SECURE—Encryption kit for 3000, 4000 and 8000 series Commodore computers. Produces 256 encryptions of single programs at random.

Cost:
Contact company

Company:
BPI Micro Systems, Ltd.
705 Progress
Scarborough, Ontario
Canada
416-431-3200

Product:
Canadian Payroll System—For the CBM 8032 computer and 8050 disk drive. Ideal for medium to larger size businesses, Canadian Payroll is designed to calculate straight payroll, overtime, and pension and medical deductions. It prints pay stubs with all payroll data for each employee, a complete payroll report after each payroll run, and a month-end report in general ledger form. It also maintains running totals of all employee pay and deductions throughout the year, and prints the contents of any employee file and a directory of employees alphabetically and numerically. A Quebec version is also available.

Cost:
Contact company

Product:
Payroll Calculator—For small business payroll calculations on CBM microcomputers. Capable of calculating payroll at regular and overtime rates, providing for additional entries such as commissions, allowing for several kinds of deductions, printing pay stubs and calculating, itemizing and printing income tax and other monthly reports. A Quebec version is also available.

Cost:
Contact company

Company:
Southern Solutions
P.O. Box P
McKinney, TX 75069

Product:
CMS System III—Accounting system for Commodore computers. Contains modules for accounts receivable, accounts payable, payroll and general ledger. The system is written so the operator can make one entry and all ledgers and journals are automatically updated to reflect that entry.

Cost:
Contact company

Product:
Business-oriented software for the Commodore 64—Includes the following:

Businessman™ general ledger system for the very small business.
The Club Manager™ organization dues management
The PayMaster™ payroll system
The BillPayer™ accounts payable
The BudgetCommittee™ home budget, check writing, bill paying
The MailMan™ mail list management
The RecordKeeper™ general data management
The FlightPlan™ aviation flight planner
The ScoreKeeper™ athletic league management
The Mailbox™ modem/file communications system
The TimeKeeper™ time management

Cost:
Contact company

USER CLUBS
Sound Off!

We're continuing to compile a list of all Commodore Users clubs throughout the country. If you'd like to add your name to the rolls, please send your club's name, address, and other pertinent information to:

Commodore Users Clubs
c/o Editor
Commodore Magazine
487 Devon Park Drive
Wayne, PA 19087

ALABAMA

Huntsville PET Users Club
9002 Berclair Road
Huntsville, AL 35802
Contact: Hal Carey
Meetings: every 2nd Thursday

ARIZONA

VIC Users Group
1206 N. Fraser Drive
Mesa, AZ 85203
Contact: Paul V. Muffuletto

Commodore User Group
Metro Computer Store
4500 E. Speedway, Suite 13
Tucson, AZ 85712
602-323-3116

ARKANSAS

Commodore/PET Users Club
Conway Middle School
Davis Street
Conway, AR 72032
Contact: Geneva Bowlin

Lawrence Hall of Science
UC Berkeley
Computer Project, Room 254
Berkeley, CA 94720
(415) 642-3598

PALS (PETS Around Livermore Society)
886 South K
Livermore, CA 94550
Contact: John Rambo

SCPUG Southern California PET Users Group
c/o Data Equipment Supply Corp.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361
Meetings: First Tuesday of each month

California VIC Users Group "VIC-VILLE"
c/o Data Equipment Supply Corp.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361
Meetings: Second Tues. of each month

Commodore Users Club
1041 Foxenwoods Drive
Santa Maria, CA 93455
(805) 937-4106
Contact: Greg Johnson

Valley Computer Club
2006 Magnolia Blvd.
Burbank, CA
(213) 849-4094
1st Wed. 6 p.m.

Valley Computer Club
1913 Booth Road
Ceres, CA 95307

PUG of Silicon Valley
22355 Rancho Ventura Road
Cupertino, CA 95014

BAMBUG
1450 53rd Street
Emeryville, CA
(415) 523-7396

North Orange County Computer Club
3030 Topaz, Apt. A
Fullerton, CA 92361
Dave Smith

Lincoln Computer Club
750 E. Yosemite
Manteca, CA 95336
John Fung, Advisor

PET on the Air
525 Crestlake Drive
San Francisco, CA 94132
Max J. Babin, secretary

PALS (PETs Around Livermore Society)
886 South K
Livermore, CA 94550
(415) 449-1084
Every third Wednesday 7:30 p.m.
Contact: J. Johnson

SPHINX
314 10th Avenue
Oakland, CA
(415) 451-6364
Every 2nd & 4th Thurs.

San Diego PUG
c/o D. Costarakis
3562 Union Street
(714) 235-7626
7 a.m.-4 p.m.

Walnut Creek PET Users Club
1815 Ygnacio Valley Road
Walnut Creek, CA 94596

Jurupa Wizards
4526 Kingsbury Pl.
Riverside, CA 92503
Contact: Walter J. Scott

PET/CBM/VIC User Club
c/o General Computer Store
22323 Sherman Way #9
Canoga Park, CA 91303
Contact: Tom Lynch

Vincent Yanniello's VIC 20 Software Exchange Club
2130 Colby Avenue
West Los Angeles, CA 90025
(213) 479-3000

The Commodore Connection
2301 Mission St.
Santa Cruz, CA 95060
408-425-8054
Bud Massey

COLORADO

VICKIMPET Users' Group
4 Waring Lane, Greenwood Village
Littleton, CO 80121
Contact: Louis Roehrs

CONNECTICUT

John F. Garbarino
Skiff Lane Masons Island
Mystic, CT 06355
(203) 536-9789

Commodore User Club
Wethersfield High School
411 Wolcott Hill Road
Wethersfield, CT 06109
Contact: Daniel G. Spaneas

VIC Users Club
c/o Edward Barszczewski
22 Tunxis Road
West Hartford, CT 06107

New London County Commodore Club
Doolittle Road
Preston, CT 06360
Contact: Dr. Walter Doolittle

FLORIDA

Jacksonville Area PET Society
401 Monument Road, #177
Jacksonville, FL 32211

Richard Prestien
6278 SW 14th Street
Miami, FL 33144

South Florida PET Users Group
Dave Young
7170 S.W. 11th
West Hollywood, FL 33023
(305) 987-6982

VIC Users Club
c/o Ray Thigpen
4071 Edgewater Drive
Orlando, FL 32804

IDAHO

GHS Computer Club
c/o Grangeville High School
910 S. D St.
Grangeville, ID 83530
Don Kissinger

ILLINOIS

Shelly Wernikoff
2731 N. Milwaukee Avenue
Chicago, IL 60647

VIC 20/64 Users Support Group
c/o David R. Tarvin
114 S. Clark Street
Pana, IL 62557
(217) 562-4568

Central Illinois PET User Group
635 Maple
Mt. Zion, IL 62549
(217) 864-5320
Contact: Jim Oldfield

ASM/TED User Group
200 S. Century
Rantoul, IL 61866
(217) 893-4577
Contact: Brant Anderson

PET VIC Club (PVC)
40 S. Lincoln
Mundelein, IL 60060
Contact: Paul Schmidt, president

Rockford Area PET User's Group
1608 Benton Street
Rockford, IL 61107

Commodore Users Club
1707 East Main St.
Olney, IL 62450
Contact: David E. Lawless

VIC Chicago Club
3822 N. Bell Ave.
Chicago, IL 60618
John L. Rosengarten

INDIANA

PET Users
Jerry Brinson
PO Box 36014
Indianapolis, IN 46236
(317) 898-3604

Cardinal Sales
6225 Coffman Road
Indianapolis, IN 46268
(317) 298-9650
Contact: Carol Wheeler

CHUG (Commodore Hardware Users Group)
12104 Meadow Lane
Oaklandon, IN 46236
Contact: Ted Powell

VIC Indy Club
P.O. Box 11543
Indianapolis, IN 46201
(317) 898-8023
Ken Ralston

IOWA

PET Users Group
c/o Don Vorhies
1321 42 St. SE
Cedar Rapids, IA 52403

Commodore User Group
114 8th St.
Ames, IA 50010

Quad City Commodore Club
1721 Grant St.
Bettendorf, IA 52722
319-355-2641
John Yigas

KANSAS

Wichita Area PET Users Group
2231 Bullinger
Wichita, Kansas 67204
(316) 838-0518
Contact: Mel Zandler

Kansas Commodore Computer Club
101 S. Burch
Olathe, KS 66061
Contact: Paul B. Howard

Commodore Users Group
6050 S. 183 St. West
Viola, KS 67149
Walter Lounsbery

LOUISIANA

Franklin Parish Computer Club
#3 Fair Ave.
Winnsboro, LA 71295
James D. Mays, Sr.

NOVA
917 Gordon St.
New Orleans, LA 70117
(504) 948-7643
Kenneth McGruder, Sr.

MARYLAND

Assoc. of Personal Computer Users
5014 Rodman Road
Bethesda, MD 20016

Blue TUSK
700 East Joppa Road
Baltimore, MD 21204
Contact: Jim Hauff

House of Commodore
8835 Satyr Hill Road
Baltimore, MD 21234
Contact: Ernest J. Fischer

MASSACHUSETTS

Eastern Massachusetts VIC Users Group
c/o Frank Ordway
7 Flagg Road
Marlboro, MA 02173

VIC Users Group
c/o Ilene Hoffman-Sholar
193 Garden St.
Needham, MA 02192

Commodore Users Club
Stoughton High School
Stoughton, MA 02072
Contact: Mike Lennon

Berkshire PET Lovers CBM Users Group
Taconic High
Pittsfield, MA 01201

The Boston Computer Society
Three Center Plaza
Boston, MA 02108
(617) 367-8080
Mary E. McCann

MICHIGAN

David Liem
14361 Warwick Street
Detroit, MI 48223

VIC Users Club
University of Michigan
School of Public Health
Ann Arbor, MI 48109
Contact: John Gannon

Toledo PETS
734 Donna Drive
Temperance, MI 48182
Contact: Gerald Carter, president

Commodore User Club
32303 Columbus Drive
Warren, MI 48093
Contact: Robert Steinbrecher

Commodore Users Group
c/o Family Computer
3947 W. 12 Mile Rd.
Berkley, MI 48072

VIC Commodore Club
2765 Bristol Rd.
Dowling, MI 49050
Bill Kelley

MINNESOTA

MUPET (Minnesota Users of PET)
P.O. Box 179
Annandale, MN 55302
c/o Jon T. Minerich

Twin Cities Commodore Computer Club
6623 Ives Lane
Maple Grove, MN 55369
(612) 424-2425
Contact: Rollie Schmidt

MISSOURI

Clearwater Club
Clearwater School
Star Route
Piedmont, MO 63957
Contact: Carolyn Polk

KCPUG
5214 Blue Ridge Boulevard
Kansas City, MO 64133
Contact: Rick West
(816) 356-2382

PET SET Club of St. Louis
1501 Carman Road
Manchester, MO 63011
(314) 527-3236
Contact: Ed or Til Keil

MONTANA

Powder River Computer Club
Powder River County High School
Broadus, MT 59317
Contact: Jim Sampson

Commodore User Club
1109 West Broadway
Butte, MT 59701
Contact: Mike McCarthy

NEVADA

Las Vegas PET Users
4884 Iron Avenue
Las Vegas, NV 89110

NEW JERSEY

Amateur Computer Group
18 Alpine Drive
Wayne, NJ 07470

Somerset Users Club
49 Marcy Street
Somerset, NJ 08873
Contact: Robert Holzer

Educators Advisory
P.O. Box 186
Medford, NJ 08055
(609) 953-1200
John Handfield

VIC-TIMES
46 Wayne St.
Edison, NJ 08817
Thomas R. Molnar

NEW HAMPSHIRE

Northern New England Computer Society
PO Box 69
Berlin, NH 03570

TBH VIC-NICs
P.O. Box 981
Salem, NH 03079

NEW MEXICO

Commodore Users Group
6212 Karlson, NE
Albuquerque, NM 87113
505-821-5812
Danny Byrne

NEW YORK

Capital District PET Users
Ben Green
Albany Area, NY
(518) 370-1820

Long Island PET Society
Ralph Bressler
Harborfields HS
Taylor Avenue
Greenlawn, NY 11740

PET User Club of Westchester
Box 1280
White Plains, NY 10602
Contact: Ben Meyer

LIVE (Long Island VIC Enthusiasts)
17 Picadilly Road
Great Neck, NY 11023
Contact: Arnold Friedman

Commodore Masters
25 Croton Ave.
Staten Island, NY 10301
Contact: Stephen Farkouh

VIC Users Club
76 Radford St.
Staten Island, NY 10314
Contact: Michael Frantz

VIC Users Club
c/o Christopher Kwasnicki
44 Harvey Ave.
Staten Island, NY 10314

PUG of Rockland County
c/o Ross Garber
14 Hillside Court
Suffern, NY 10901
(914) 354-7439

West Chester County VIC Users Group
P.O. Box 146
Pelham, NY 10552
Joe Brown

SPUG
4782 Boston Post Rd.
Pelham, NY 10803
Paul Skipski

VIC 20 User Club
151-28 22nd Ave.
Whitestone, NY 11357
Jean F. Coppola

VIC 20 User Club
339 Park Ave.
Babylon, NY 11702
(516) 669-9126
Gary Overman

VIC User Group
1250 Ocean Ave.
Brooklyn, NY 11230
212-859-3030
Dr. Levitt

NORTH CAROLINA

Amateur Radio PET User's Group
PO Box 30694
Raleigh, NC 27622
Contact: Hank Roth

Commodore Users Club
4241 Castleton Road
Charlotte, NC 28211
Contact: Ed Harris

VIC Users Club
c/o David C. Fonenberry
Route 3, Box 351
Lincolnton, NC 28092

Microcomputer Users Club
Box 17142 Bethabara Sta.
Winston-Salem, NC 27116
Joel D. Brown

VIC Users Club
Rt. 11, Box 686
Hickory, NC 28601
Tim Gromlovits

OHIO

Dayton Area PET User Group
933 Livingston Drive
Xenia, OH 45385
B. Worby, president
(513) 848-2065
J. Watson, secretary
(513) 372-2052

Central Ohio PET Users Group
107 S. Westmoor Avenue
Columbus, Ohio 43204
(614) 274-6451
Contact: Philip H. Lynch

OKLAHOMA

Southwest Oklahoma Computer Club
4311 Floyd Ave. NW
Lawton, OK 73505
Joe Teeples

NW PET Users Group
John F. Jones
2134 N.E. 45th Avenue
Portland, OR 97213

PENNSYLVANIA

PET User Group
Gene Beals
PO Box 371
Montgomeryville, PA 18936

Penn Conference Computer Club
c/o Penn Conference of SDA
720 Museum Road
Reading, PA 19611
Contact: Dan R. Knepp

PACS PET Users Group
20th & Olney Streets
Philadelphia, PA

Glen Schwartz
807 Avon
Philadelphia, PA 19116

Gene Planchak
4820 Anne Lane
Sharpsville, PA 15150
(412) 962-9682

PPG (Pittsburgh PET Group)
c/o Joel A. Casar, DMD
2015 Garrick Drive
Pittsburgh, PA 15235
(412) 371-2882

Bucks County Commodore User Club
72 E. Rambler Dr.
Holland, PA 18966
(215) 322-0394
Jim Dubrouillet

PUERTO RICO

CUG of Puerto Rico
1607 Calle Colon
Santurce, PR 00911
Ken Burch

VIC 20 User Group
655 Hernandez St.
Miramar, PR 00907
Robert Morales, Jr.

SOUTH DAKOTA

PET User Group
515 South Duff
Mitchell, SD 57301
(605) 996-8277
Contact: Jim Dallas

TENNESSEE

River City Computer Hobbyists
Memphis, TN
1st Mon. at Main Library

Nashville VIC Users Group
American Computer Store
1004 8th Ave., S.
Nashville, TN 37203
(615) 242-8592
Jane Maggard
1st Thurs. of month, 6 pm

Commodore User Club
Metro Computer Center
1800 Dayton Blvd.
Chattanooga, TN 37405
Mondays 7:30 pm

TEXAS

SCOPE
1020 Summit Circle
Carrolton, TX 75006

PET Users
2001 Bryan Tower
Suite 3800
Dallas, TX 75201

Larry Williams
PO Box 652
San Antonio, TX 78293

PET User Group
John Bowen
Texas A & M Microcomputer Club
Texas A & M, TX

CHUG (Commodore Houston Users Group)
8738 Wildforest
Houston, TX 77088
(713) 999-3650
Contact: John Walker

Corpus Christi Commodores
3650 Topeka St.
Corpus Christi, TX 78411
512-852-7665
Bob McKelvy

UTAH

Utah PUG
Jack Fleck
2236 Washington Blvd.
Ogden, UT 84401

The Commodore User's Club
742 Taylor Avenue
Ogden, Utah 84404
Contact: Todd Woods Kap, president;
David J. Shreeve, vice president

The VIClic
799 Ponderosa Drive
Sandy, UT 84070
Contact: Steve Graham

VIC 20 Users
324 N. 300 W.
Smithfield, UT 84335
Dave DeCorso

Northern Utah VIC & 64 Users Group
P.O. Box 533
Garland, UT 84312
David Sanders

VIRGINIA

Northern VA PET Users
Bob Karpen
2045 Eakins Court
Reston, VA 22091
(803) 860-9116

VIC Users Group
Rt. 2, Box 180
Lynchburg, VA 24501
Contact: Dick Rossignol

VIC Users Group
c/o Donnie L. Thompson
1502 Harvard Rd.
Richmond, VA 23226

Dale City Commodore User Group
14752 Danville Rd.
Dale City, VA 22193
(703) 680-2270
James Hogler

Tidewater Commodore Users Group
417 Bethune Dr.
Virginia Beach, VA 23452
H. Thomas Baise. III

WASHINGTON

NW PET Users Group
2565 Dexter N. 3203
Seattle, WA 98109
Contact: Richard Ball

PET Users Group
c/o Kenneth Tong
1800 Taylor Ave. N102
Seattle, WA 98102

VIC 20 Computer Club
947 N. Burroughs Ave.
Oak Harbor, WA 98277
Michael D. Clark

User Bulletin Board

USER GROUPS FORMING

Florida
VIC User Group forming:
Contact Bob Kirksey
7624 Las Vegas Lane
Pensacola, FL 32504

Georgia
Commodore User Group forming:
Contact Bruce Godley
615 Northway Lane
Atlanta, GA 30342
404-252-9890

Illinois
VIC International Computer Knowledge Information Exchange (VICKIE): International clearinghouse and newsletter devoted to VIC 20 and Commodore 64 now forming. Membership $20/year, $17.50 for students. Contact John L. Rosengarten, President
VICKIE
3822 N. Bell Ave.
Chicago, IL 60618

SUPERPET NEWSLETTER

SuperPETters who'd like to keep informed on more of the latest SuperPET developments can get a free 10-page monthly newsletter. Send a 20-cent stamp (no envelope, just a stamp) to:
The Editor
SuperPET Gazette
P.O. Box 411
Hatteras, NC 27943

They're also looking for an associate editor, by the way. Contact same.

User Clubs Continued

WISCONSIN

Sewpus
c/o Theodore J. Polozynski
PO Box 21851
Milwaukee, WI 53221

Waukesha Area Commodore User Group (WACUG)
256½ W. Broadway
Waukesha, WI 53186
Contact: Walter Sadler
(414) 547-9391

Commodore User Group
1130 Elm Grove St.
Elm Grove, WI 53122
Tony Hunter

CANADA

Toronto PET Users Group
381 Lawrence Ave. West
Toronto, Ontario, Canada
M5M 1B9
(416) 782-9252
Contact: Chris Bennett

PET Users Club
c/o Daniel Cayer
R.R. 6
Simcoe, Ontario
Canada N3Y 4K5

Vancouver PET Users Group
Box 91164
West Vancouver, British Columbia
Canada V7V 3N6

CCCC (Canadian Commodore Computer Club)
c/o Strictly Commodore
47 Coachwood Place
Calgary, Alberta, Canada
T3H 1E1
Contact: Roger Olanson

W.P.U.G.
9-300 Enniskillen Ave.
Winnipeg, Manitoba R2V 0H9
Larry Neufeld

VIC-TIMS
2-830 Helena St.
Trail, British Columbia
VIR 3X2
604-368-9970
Greg Goss

KOREA

Commodore Users Club
K.P.O. Box 1437
Seoul, Korea
Contact: S. K. Cha

MEXICO

Asociacion De Usuarios Commodore
c/o Alejandro Lopez Arechiga
Holbein 174-6° Piso
Mexico 18, D.F.

Club de Usarios Commodore
Sigma del Norte
Mol del Valle, Local 44
Garza Garcia, N.L. 66220

NEW ZEALAND

Commodore Users Group
Meet at VHF Clubrooms
Hazel Ave.
Mount Roskill
3rd Wed of month, 7:30 pm
Roger Altena 278-5262

NORWAY

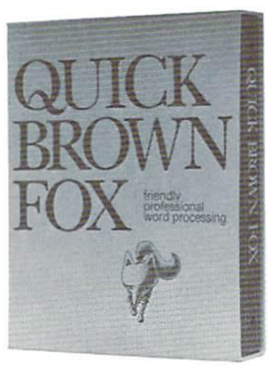
VIC Club of Norway
Nedre Bankegt 10,
1750 Halden
Norway

Ad designed by Todd Lerner

WORD PROCESSOR?

You bet! Quick Brown Fox word processing software has more features than Word Star and runs on your VIC or Commodore 64. And it can grow and grow. Add memory, 80 column display, disks, even a letter quality printer. We'll show you how a 1st-class word processor can be yours for less than $2000! You can even have a "student" system for less than $700!

It all starts with the Quick Brown Fox at $65.



Call or write for our free brochure
Quick Brown Fox 548 Broadway New York NY 10012 (800) 547-5995 Ext 194
Dealer Inquiries Invited (212) 925-8290

Our old friend Captain Scuttlebutt has been lurking in the shadows again, picking up all the latest rumors on Commodore's not-yet-offically-mentionable plans. Since he obviously has nothing better to do, we've asked him to jot down some of his most recent gleanings from the Commodore grapevine. But remember — we official types have nothing to do with any of this, and will deny it all if questioned.

I got wind, recently, of a hot poker game going on near the Commodore offices every Tuesday night. Rumor had it that each of the players in the game owned a different kind of home computer, one of which was a VIC 20. The thought of it made my prodigious aural cavities tingle in anticipation.

I hightailed it over there one evening before the players had arrived, and stationed myself inconspicuously behind a large Australian umbrella tree in the corner. I was delighted, of course, when, after the second hand was dealt, they began to talk about their computers. And I was overjoyed to discover, as the conversation progressed, that I had most definitely chosen the right computer company in which to ply my disreputable trade. From what I overheard about those other poor benighted computer companies, I quickly concluded that Commodore is without a doubt the real newsmaker and status-quo-shaker in the business — which is exactly where we rumor mongers thrive. Lots of things to get our ears into, so to speak.

But back to the poker game. Let me tell you what I found out. Did you know, for instance, that some other home computer companies charge extra for a disk drive controller — on top of what they charge for the disk drive itself, which is already a pretty penny? That's what they said that night.

Captain Scuttlebutt's Unbelievable Rumors

A certain company based in the state known for yellow roses, for instance, sells its disk drive for $399 (about the same as the VIC 20 disk drive) BUT to make the silly thing work (which is always a nice touch), you've got to shell out an additional $299 for a controller. Not only that, but they said it uses some of the computer's RAM to run it. A couple of other companies also have stupid drives, as well, I heard — some stupider than others. Nothing like our highly intelligent VIC peripherals that leave the RAM unscathed. That's what they said. If you don't believe it, you can sit in that corner yourself some night. Not pleasant after a few hours.

Before we leave the topic of disk drives, let me also mention that, according to what I heard that night behind the umbrella tree, the VIC disk drive has a larger capacity than most — 174,000 bytes to be exact. The poor fellow who owns that fruit company's very expensive computer system was not happy to hear this, since his (very expensive) drive has only a 143,000 byte capacity.

Modems, as you might expect, were another volatile topic that night. We all know the VICMODEM costs only $109.95, and has the software and interface included. Imagine the shock of the other computer owners, whose modems cost upwards of $200 — and then need an interface that costs an additional — are you ready? — $200 plus. I know I've got the figures straight. I was taking notes on my cuff. And not only that, but the VIC-MODEM plugs directly into the phone line, so you don't get background noise. I began to feel sorry for the fellows who got stuck with those old fashioned acoustic couplers — especially at those horrific prices.

Or take something as absolutely essential as BASIC language built into the computer—something I know VIC owners take for granted. I almost knocked over the umbrella tree when I heard that some other companies charge extra for an equivalent BASIC! And the players argued and argued over their BASIC's speed of execution—but it kept coming up the same. VIC's highlevel Microsoft BASIC simply runs faster. And allows more flexibility in programming. And has an unlimited number of dimensions per array. And talks English. And has a real-time clock. What can I say? These fellows were all quite expert. I had to believe them.

Printers? They got into it over that, too. The VIC has a real one—an 80 column, tractor-feed, dot-matrix impact printer. It was hard for the Yellow Rose and the grown-up-game-machine to comprehend what our VIC owner (who shall remain nameless) was talking about. One of them uses an old thermal printer with specially treated paper on a roll. The other one's printer looks more like a large adding machine, with 40 columns and paper on a roll. Would I kid you? I heard all this.

And those memory expander cartridges we all have grown to know and love? I'll bet there are people out there right now who can't imagine buying a computer that couldn't be expanded easily. Yet, from what I heard that night, tucked away in my corner, I can safely say that there are, indeed, computers that can be expanded only through expensive tinkering that, in ▶

some cases, voids the warranty. That's what they said.

Keyboards? We're all used to that good-feeling typewriter keyboard on the VIC—but not every company thought of that when they were putting their computers together. Graphic characters? Color controls? Programmable function keys? If you ever get into a discussion with someone who owns a different computer, I think you'll find out you've been spoiled. Our friend the VIC owner said that very thing, in fact—several times that night.

I could, of course, tell you even more about what I heard that night from my secret outpost behind the umbrella tree—about cassette decks and monitors and synthesizers (our saddened fruit company computer owner laid out as much money just for a synthesizer as the cost of a Commodore 64, which comes equipped with a synthesizer). But why kick the competition when it's down?

So, let's leave the poker game behind (how I extricated myself from my hide-away is a whole story in itself) and go on to even more exciting and uplifting rumors. I've been dying to tell you, for instance, about something I've heard them calling the PET 64. My understanding, listening from behind the partition, is that it's a Commodore 64 with a built-in monochromatic screen, and it's intended mainly for educational use. I actually saw a picture of it in Commorore's (very nicely done) Annual Report. Who would have thought that, beneath that homey PET exterior beats the heart of a sophisticated 64, complete with music synthesizer?

Not to drag this on too long, but I just can't resist getting it all off my chest. You've probably heard, by now, about LOGO and Pilot—two languages begin used to teach children programming, and, among other things, great ways to create graphics. Team them up with the Commodore 64, and, believe me, you've got a super package. Which is exactly what I've heard Commodore is doing. Don't quote me on dates, but you should be seeing one or both of these in the near future. If I'm very lucky, I'll be able to corner Softwareman, another Commodore misfit who hangs around the offices here—but never ventures very far from the telephone booth—and get the word on when we'll be seeing these and other new developments for the 64.

Signing off for now. Keep your ear to the ground—but always make sure your head is attached to it.

Circle #47 on the Reader Service Card

PROGRAMMING a **VIC-20?**
Beginner or Expert.

PAL®

Programmers **A**ids
and **L**ogs

Can Help YOU!

Look what you get!

- **EZ KEY** Quick Guide to all keys, pokes, reverses, CHR$, set 1 - set 2
- **FULL COLOR** color combination chart
- **EZ GRAPH** graphics programming aid
- **EZ NOTE** sound music worksheets
- **LOTS** of tear-out SCREEN LAYOUT forms and various programming forms and worksheets
- ***BASIC*-LY EZ** condensed basic dictionary
- **SOFTWARE & HINTS** Log Sheets
- **TAPE CASSETTE** Log Book and **MORE!**

EVERYTHING YOU NEED FOR
EZ Programming!

Send **$9.95** + $2.00 shipping (CA res. add 6% tax)
Check, M.O., NO C.O.D.'s to:

PM PRODUCTS
4455 Torrance Blvd., #177, Torrance, CA 90503

satisfaction
guaranteed

☆ dealer inquiries invited ☆

Circle #48 on the Reader Service Card

UNDER $5^{00}
FOR THE VIC 20

Personalized Birthday Program (5K)

Surprise your friends on their birthday party with this cute program, complete with the song, their name, the cake with candles and a magic word

$3.95

Cogno (8K)

An exciting game of shape recognition. Several skill levels. Fun and sound in color (joystick or keyboard)

$4.95

Send check to:

Soft 4 You
P.O. Box 3254, Reston, Va. 22090

Soft 4 You:
Our prices are cheap, not our programs

Circle #49 on the Reader Service Card

COMPU SENSE

CARDRITE

A graphic
light pen for
the VIC-20®.

Supplied with
a cassette
of six programs.

$29.95

TO ORDER:
P. O. BOX 18765
WICHITA, KS 67218
(316) 684-4660

Personal checks accepted
(Allow 3 weeks) or
C.O.D. (Add $2)
Handling charges $2.00

* VIC-20 is a registered trademark of Commodore, Inc.

MasterCard VISA

VIC-TAX VIC-TAX VIC-TAX VIC-TAX VIC-TAX

AND ANY Vic-20®

COMPUTER

TO PREPARE YOUR INCOME TAX

VicTax WILL RUN ON ANY Vic-20 COMPUTER

VicTax WILL SAVE TIME AND IMPROVE ACCURACY OF TAX PREPARATION

VicTax WAS DEVELOPED BY AN EXPERIENCED ACCOUNTING FIRM

VicTax IS USER FRIENDLY SOFTWARE WITH A DETAILED MANUAL

VicTax IS THE FINEST INCOME TAX SOFTWARE AVAILABLE FOR Vic-20

Circle #50 on the Reader Service Card

CASSETTE AND DETAILED MANUAL $19.95

AVAILABLE FROM YOUR LOCAL DEALER OR FROM NORTHLAND ACCOUNTING

NORTHLAND ACCOUNTING, INC.

606 SECOND AVENUE
TWO HARBORS, MN 55616
(218) 834-3600

VISA master charge

VIC-TAX VIC-TAX VIC-TAX VIC-TAX VIC-TAX

MICROSPEC

Quit Playing Games . . .

Disk Based Software to Make Your Computer Get Down to Business

Disk Based Data Manager—Create and manage your own data base. Allows you to create, add, change, delete, search, sort, print, etc. Available for VIC-20, Commodore 64, any CBM or Pet.
VIC-20 59.95 All others 79.95

Inventory Control Manager—Fast, efficient inventory package which will manage your day to day inventory requirements. Provides information on sales and movement of items.

Mailing List Manager—4,050 items per 8050 disk, 1,300 on 4040 disk and 1,200 on 1540/1541 disk. User defined label format (1—4) across.

Payroll System—Full featured complete Payroll System. Up to 350 employees on a 8050 disk. Prints checks, 941's and W-2's. For the CBM 8032/8050, 4032/4040, Commodore 64/1541.

Hospitality Payroll—The most complete payroll system written specifically for the Restaurant Industry available today. Recognizes tip and meal credits, pay advances, salaried and hourly employees, etc. For the CBM 8032/8050.

CONTACT US FOR ALL YOUR DISK BASED SOFTWARE NEEDS

Call for specifics on Hardware Configurations.

Send Self-Addressed Stamped Envelope for Catalogue of Games and other Applications

DEALER INQUIRIES WELCOME

2905 Ports O'Call Court
Plano, Texas 75075
(214) 867-1333

VISA MasterCard

VISA and MASTERCARD Accepted

Circle #51 on the Reader Service Card

Q&A HOTLINE

Q: Can I use my VIC 20 when I go to another country?

A: Commodore does not recommend using U.S. VIC's internationally. You may run into problems with local power and broadcast requirements.

The video output of a VIC 20 from the U.S. conforms to the NTSC standard, as does an American TV set. Many other parts of the world use the PAL standard, which is not compatible with NTSC. If you bring along your own TV set and transformer, you'll be safe.

The countries that use the NTSC standards are Barbados, Bermuda, Canada, Chile, Colombia, Cost Rica, Curaco, Dominican Republic, Ecuador, El Salvador, Guam, Guatemala, Honduras, Japan, South Korea, Mexico, Netherlands Antilles, Nicaragua, Panama, Peru, Phillippines, St. Kitts, Surinam, Taiwan, Trinidad, Tobago, and the Virgin Islands.

Q: What do I do if my VIC is defective?

A: Once you've checked the trouble-shooting chart in the manual, bring the computer back to the place of purchase with your receipt. They will repair or replace the unit.

In extreme situations, you can forward the computer to Commodore Business Machines, 390 Reed Street, Santa Clara, CA 95050. Be sure to include your proof-of-purchase or you will be charged for the repair.

After the warranty expires, send the computer to your nearest Commodore service center; either the Santa Clara address above or to Commodore Business Machines, Service Department, 950 Ariport Road, West Chester, PA 19380.

Out-of-warranty repair charges are:

VIC 20	$55
Datasette	$35
VIC 1515 or 1525	$75
VIC 1540 or 1541	$85
VIC 1210 3K expander	$20
VIC 1110 8K expander	$30
VIC 1111 16K expander	$55
VIC 1600 modem	$60

Q: Where do I get commands like MERGE, RENUMBER, and TRACE?

A: There is a VIC cartridge called *Programmers Aid* containing debugging and editing commands: AUTO for creating line numbers automatically, RENUMBER to change program line numbers, DELETE to erase program lines, FIND to search for characters within a program, CHANGE to find and modify characters within a program, MERGE to mix two programs together, DUMP to display all variables and their values, HELP to locate program errors, TRACE to display line numbers as they execute, STEP to execute a program one line at a time, and KEY to program function keys.

For PET and CBM owners, these commands are available on various ROM products, like the Programmer's Toolkit from Palo Alto IC's.

That Does Not Compute...

Using the Joystick on the PET

by Elizabeth Deal, October/November COMMODORE, pages 68-70:

With some embarassment we're re-running the entire program listing, which had been printed out on a printer that used non-standard symbols, and somehow managed to get through in that condition. Apologies.

```
100 REM--------------------------------
110 REM JOYSTICK/KEYBOARD DRAWING
120 REM ELIZABETH DEAL
130 REM--------------------------------
140 IFAA=0THENGOSUB430:GOSUB560:AA=1
150 IV=5:X1=XM:Y1=YM:DD=3:CL=SC(INT(Y1/2)):CV=DM(5):POKECL,CV
160 Q=0:KY=5:PK=5:AV=0:SP=32:AN=48:KX=255:K5=5:K1=1:K9=9:K$=CHR$(0)
170 REM---JS/KB LOOP
180 IV=JS%(PEEK(JS)ANDJ1):IFIV<>K5THENKY=IV:GOTO220
190 GETI$:IFI$=""THENIFPEEK(KM)<>KXTHENKY=PK:GOTO220
200 IV=ASC(I$+K$):KY=IV-AN:IFKY<K1ORKY>K9THENGOSUB300:IFEGOTO180
210 IFQTHENRETURN:<END>
220 POKECL,CV:X=X1+XX(KY):Y=Y1+YY(KY)
230 IFX<XMORX>XLORY<YMORY>YLTHENX=X1:Y=Y1:GOTO280
240 CL=SC(Y%(Y))+X%(X):V1=PEEK(CL):IFAVTHENIFV1<>SPTHENIFI(V1)=QGOTO280
250 SQ=AM(XANDAM,YANDAM):ONDDGOTO260,270,280
260 POKECL,C(I(V1)ORSQ):GOTO280
270 POKECL,C(I(V1)ANDNOTSQ)
280 CV=PEEK(CL):POKECL,DM(PK):PK=KY:X1=X:Y1=Y:GOTO180
290 REM---USR,QUIT,CLS,DRAW,ERASE,JUMP,RESET SKIP MODE
300 E=1:IF(IVAND127)=94THENGOSUB600:RETURN
310 IFIV=82THENAV=1-AV:POKESE,DM(Z1-AV):RETURN
320 IFIV=81THENE=0:Q=1:RETURN
330 IFIV=147THENGOSUB380:PRINTI$;:CV=32:DD=3:X1=XM:Y1=YM:RETURN
340 CM=DD:DD=IV-67:IFIV=74THENDD=3
350 IFDD<0ORDD>3THENDD=CM:PK=5
360 POKESE-2,DM(DD+Z2):RETURN
370 REM---CONFIRM CLS
380 V=PEEK(SC):POKESC,191
390 GETI$:IFI$=""GOTO390
400 IFASC(I$)<>IVTHENI$=K$
410 POKESC,V:RETURN
420 REM---SETUP1
430 SC=32768:SS=SC+1024:V1=PEEK(SS):SW=80:POKESS,96:IFPEEK(SC)=96THENSW=40
440 POKESS,V1:BL=SC+24*SW:SE=BL+SW-1:KM=15 :IFPEEK(50003)=0THENKM=525
450 POKE59459,0:JS=59471:J1=15:J2=1:JJ$="5550579851325465"
460 DIM JS%(15):FORJ=1TO16:JS%(J-1)=ASC(MID$(JJ$,J))-48:NEXTJ
470 FORI=1TO3:Q=I-2:FORJ=0TO6STEP3:XX(I+J)=Q:NEXTJ,I
480 FORI=1TO9:YY(I)=INT((I-1)/3)-1:NEXTI
490 DIM SC(24),X%(SW*2),Y%(49):FORJ= TO24:SC(J)=BL-J*SW:NEXTJ
500 FORJ=0TO2*SW-1:X%(J)=J/2:NEXTJ:FORJ=0TO49:Y%(J)=J/2:NEXTJ
510 Z1=14:Z2=9:DIM DM(Z1):FORJ=1TOZ1:READDM(J):NEXTJ
520 DATA 76,33,122,60,43,62,79,30,80,4,5,10,19,14
530 XL=2*SW-1:YL=49:XM=0:YM=2
540 POKE59468,12:PRINTCHR$(142)CHR$(142);:RETURN
550 REM---SETUP2, PAUL HIGGINBOTTOM
560 DIM I(255),C(15),AM(1,1):AM=1:FORI=0TO15:READC(I):I(C(I)) I:NEXT
570 FORI=0TO1:FORJ=0TO1:AM(J,I)=(J+1)*4↑I:NEXTJ,I:RETURN
580 DATA 32,123,108,98,126,97,127,252,124,255,225,254,226,236,251,160
590 REM---SAVE/DUMP/NORMAL GRAPHICS/OTHER UTILITIES
600 RETURN: NOTHING HERE
610 REM--------------------------------
620 REM JOYSTICK  #1 J1= 15 J2= 1
630 REM           #2 J1=240 J2=16
640 REM IV=JS(((PEEK(JS)ANDJ1))/J2)
650 REM--------------------------------

READY.
```

PROJECTIONS & REFLECTIONS

I would like to start off this month's issue by thanking all of the people who have responded to our software search by sending in their products for review. Please keep it coming! But, keep in mind that we are more interested in simple, personal, software products than the larger business or scientific systems.

It has come to my attention through some of the letters our readers have sent in that few people really realize just what goes into the publishing of a software product. So, this month I would like to outline the steps that we generally go through in putting a product together for distribution. Bear in mind that these tasks are actually performed in a different order, but that is usually by necessity rather than choice.

1. *A software product is brought to our attention either by being sent in (solicited or unsolicited), through our participation in its design, or by having it written in-house.*

2. *The product is reviewed by our staff. This review contains several criteria that the product must meet. It is looked at from a technical point of view as well as from a personal-appeal viewpoint. After this information is compiled I will personally review the product, as well as the comments from the reviewers. The product is then recommended for the next step or it is put in the 'thanks, but no thanks' category. If the product requires further effort we establish a dialogue with the vendor to see just what type of business arrangements may be made.*

3. *The business opportunities for vendors at this point are really two: The product can become either a Commodore product or a Commodore Approved Product. If the product is really exciting and there are no support or business reasons not to pursue a license agreement with the vendor, we try to make the product into a Commodore product. This means that we take over the entire responsibility of manufacturing, initial support, distribution and pricing. The vendor is then normally paid a percentage of each sale (unless Commodore has bought the product outright).*

If the product is exciting, but there are some support or business considerations that prevent it from becoming a Commodore product we help the vendor by making it an Approved Product. This means that we have seen the product, tested it, and agree that it should be given some special consideration by our dealers as a sales tool. It does not mean that we represent the vendor or stand behind the product financially. For the Approved Product vendor, there are a number of advantages that will help his business to grow and at the same time help Commodore sell more hardware by having good software in the marketplace. Over a period of time, our dealers have been conditioned to look for the Commodore Approved logo on a product . . . for that little extra assurance that it is a good product.

All of the above steps take considerable time and effort on our part to make sure that the product is ready for distribution. Just the preparation of manuals, from initial typing to printing, sometimes takes several weeks. Through our efforts here, we have brought several good software products to the marketplace, and will continue to do so in the future.

With your help, by supplying us with good products and readable documentation, we can generate more products faster. So, please keep the products coming, and if you have already sent in a product please be patient.

Think Software.

Paul Goheen
Software Products Manager

Advertisers' Index

commodore

Your VIC 20 never had it so good!

VIC 20 owners rejoice! HES presents a complete range of software from our exciting series of games to our professional group of utility and language programs.

Our new cartridge programs include: HES MON, an indispensible monitor for assembly language programmers; HES WRITER, a word processing program; Turtle Graphics, a fun and easy way to learn computer programming; and VIC FORTH, a powerful language that is many times faster than BASIC, yet easier to use than assembly language.

HES is committed to offering high-quality, well-documented computer programs on a continual basis. Look for our cartridge and cassette based software at your local dealer.

HES

Human Engineered Software
71 Park Lane
Brisbane, California 94005
Telephone 415-468-4110

Circle #54 on the Reader Service Card